MOSTEK®

MICROCOMPUTER SYSTEMS

Operations Manual

VAB-2 VIDEO ADAPTER BOARD

OPERATIONS MANUAL

FOR

VIDEO ADAPTER BOARD - 2

VAB - 2
(MK 79052)

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

LIST OF FIGURES

LIST OF TABLES

SECTION 1

GENERAL INFORMATION

1-1. INTRODUCTION.

1-2. The VAB-2 is an MK3870 single chip microcomputer based video terminal system on a single printed circuit board. The VAB-2 functions as an interface between a 20 mA full duplex serial data current loop, an ASCII encoded keyboard and an EIA standard video monitor.

1-3. FUNCTIONAL DESCRIPTION.

1-4. The VAB-2 is the major functional block in a complete CRT terminal, as illustrated in Figure 1-1. Communication with the external data processing equipment is via a 20 mA, full duplex current loop data link. Either ASCII or Baudot data codes may be used, each of which at two selectable data rates. Data is output as composite video for display on a CRT monitor, and operator response is input from an encoded keyboard. The keyboard transmitter and the screen receiver operate as totally independent functional units.

1-5. ELECTRICAL DESCRIPTION.

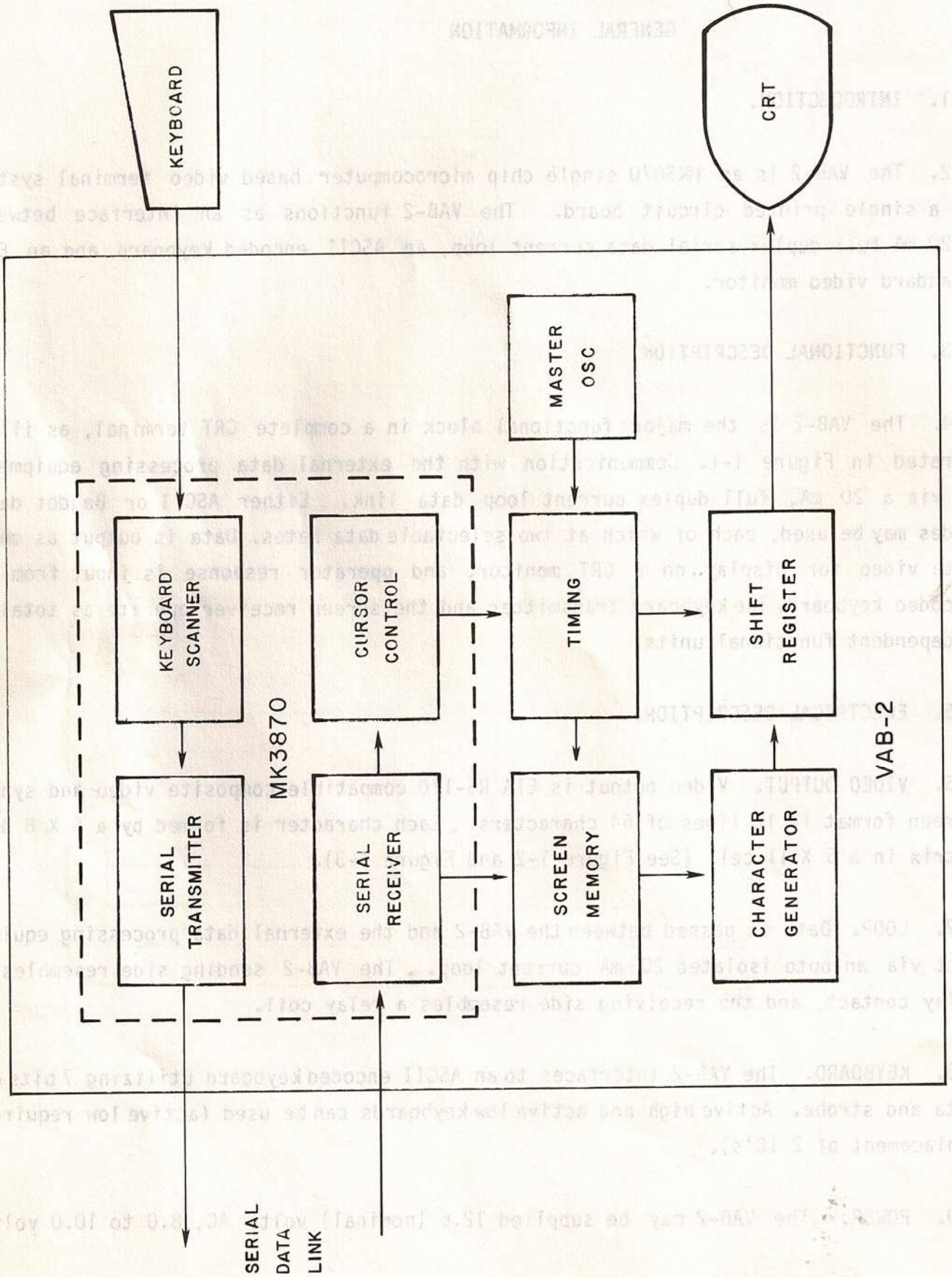
1-6. VIDEO OUTPUT. Video output is EIA RS-170 compatible composite video and sync. Screen format is 16 lines of 64 characters. Each character is formed by a 5 X 8 dot matrix in a 6 X 11 cell (See Figure 1-2 and Figure 1-3).

1-7. LOOP. Data is passed between the VAB-2 and the external data processing equipment via an opto isolated 20 mA current loop. The VAB-2 sending side resembles a relay contact, and the receiving side resembles a relay coil.

1-8. KEYBOARD. The VAB-2 interfaces to an ASCII encoded keyboard utilizing 7 bits of data and strobe. Active high and active low keyboards can be used (active low requires replacement of 2 IC's).

1-9. POWER. The VAB-2 may be supplied 12.6 (nominal) volts AC, 8.0 to 10.0 volts

Figure 1-1. System Functional Diagram.



L99992

Figure 1-2. Video Format.

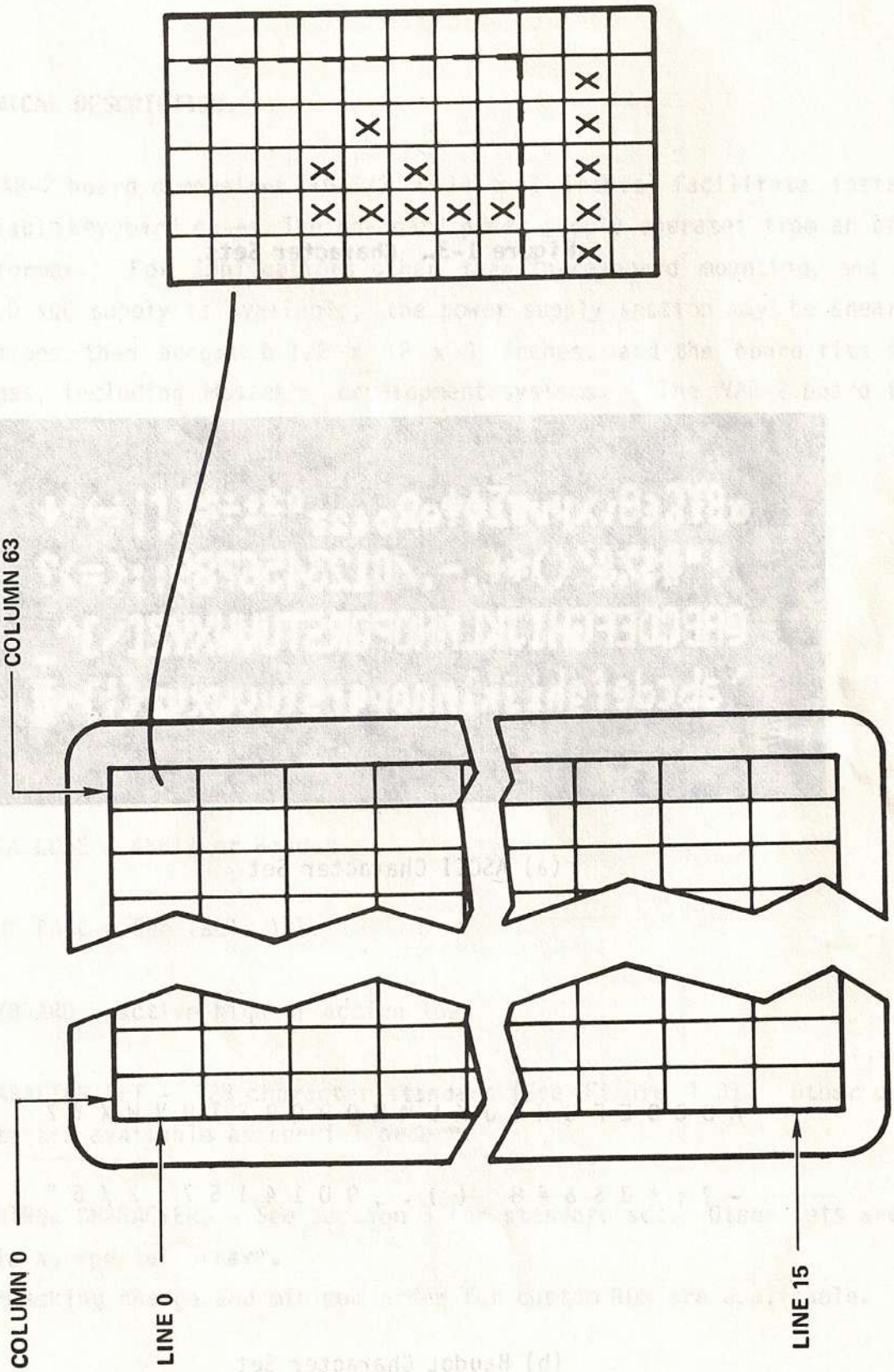
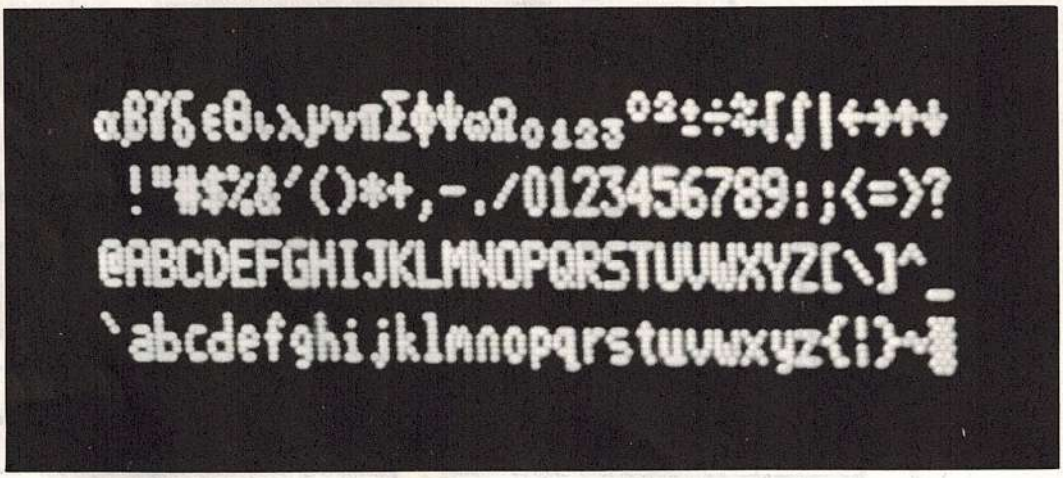


Figure 1-3. Character Sets.



(a) ASCII Character Set

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- ? : * 3 $ & # 8 ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 "

(b) Baudot Character Set

unregulated DC, or 5 volts regulated DC. An on board regulator is available when required.

1-10. **MECHANICAL DESCRIPTION.**

1-11. The VAB-2 board dimensions (6 1/2 x 14 x 1 inches) facilitate installation inside available keyboard cases. The on-board power supply operates from an off-board 12VAC transformer. For applications other than in-keyboard mounting, and where a regulated 5.0 VDC supply is available, the power supply section may be sheared off. Board dimensions then become 6 1/2 x 12 x 1 inches, and the board fits standard 12" card cages, including Mostek's development systems. The VAB-2 board is shown in Figure 1-4.

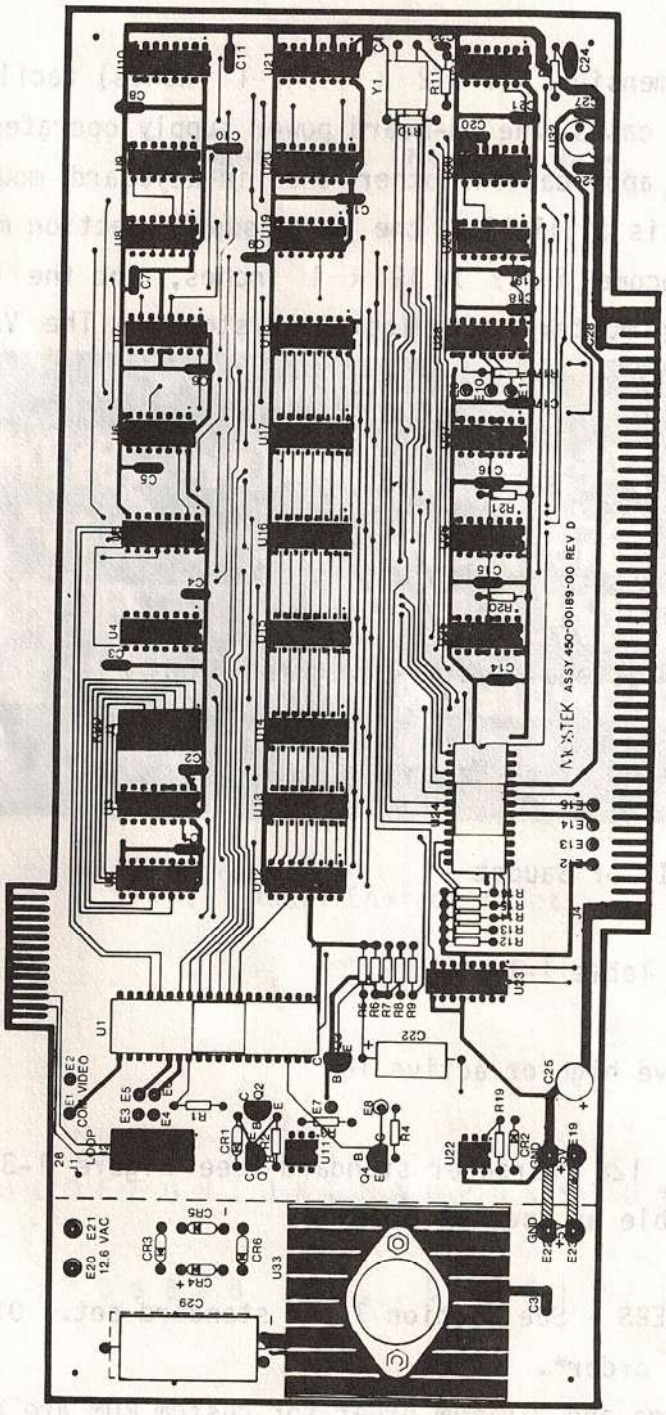
1-12. **AVAILABLE OPTIONS.**

1-13. The following options are available on the VAB-2:

1. POWER SUPPLY - AC, regulated or unregulated DC
2. LINE FREQUENCY - 50 or 60 Hz
3. DATA CODE - ASCII or Baudot
4. BAUD RATE - See Table 1-1.
5. KEYBOARD - active high or active low
6. CHARACTER SET - 128 character standard (See Figure 1-3). Other character sets are available as special order*.
7. CONTROL CHARACTERS - See Section 3 for standard set. Other sets are available as special order*.

*Masking charge and minimum order for custom ROM are applicable.

Figure 1-4. VAB-2 Circuit Board.



L99990

1-14. SPECIFICATIONS SUMMARY.

Table 1-1. Specifications.

BOARD SIZE: 6.5" x 12" x 1" without power supply
6.5" x 14" x 1" with power supply

POWER REQUIREMENTS: 8.0 to 13.0 VAC rms @ 1.0 A max*
8.0 to 10. VDC @ 1.0 A max*
5.0 VDC ± 5% @ 1.0 A max*
(* including keyboard)

VIDEO OUTPUT: 1.5 Vp-p into 75 Ω (EIA RS-170)

KEYBOARD INPUT: CODE: ASCII
HIGH: 2.8 to 5 V @ 40 µA
LOW: 0 to 0.4 V @ 1.6 mA
LOGIC: Standard TTL, positive logic**
STROBE WIDTH: 1.0 ms min.
CONNECTOR: 16 pin DIP
POWER: 5 VDC ± 5% @ 0.35 A max
(** negative logic optional)

LOOP: 20 mA nominal (15 mA min, 30 mA max) mark current, external loop supply isolated 250 V max loop to ground.

BAUD RATES: 300, 110 ASCII
74.2, 45.45 Baudot

AUX OUTPUT: HIGH: 2.4 to 5 V @ 100 µA
LOW: 0 to 0.4 V @ 1.8 mA

SECTION 2

INSTALLATION

2-1. INTRODUCTION.

2-2. VAB-2 installation options are individually discussed below, followed by a sample complete configuration shown in Figure 2-10.

2-3. UNPACKING.

2-4. The VAB-2 is shipped in a conductive plastic bag, which should be saved for subsequent handling, storage, or shipping. The integrated circuits in this assembly are high impedance MOS devices. Internal circuitry is included in each device to protect the inputs against damage due to static voltage; however, the assembly should remain in the conductive bag until ready for installation.

2-5. EXTERNAL EQUIPMENT REQUIRED (Not supplied).

2-6. VIDEO MONITOR. The VAB-2 operates with standard video monitors utilizing EIA RS-170 composite video input, such as:

SC Electronics 10M915

2-7. KEYBOARD. The VAB-2 operates with standard encoded ASCII keyboards, such as:

Cherry B70-54AA (keyboard and case)
Cherry B70-4753 (keyboard only)
Keytronics 65-1648-00 (keyboard only)

2-8. TRANSFORMER. Suitable line transformers for the VAB-2 include:

Stancor P-8384
Triad F-25X

NOTE The preceeding does not constitute a product endorsement for any of the above mentioned products. Equipment manufactured by companies other than Mostek is is subject to change of specification at any time and without notice.

2-9. MOUNTING.

2-10. NON CARD CAGE. For non card cage applications, the VAB-2 may be mounted on standoffs at the four board corners, and at two interior mounting holes. Care should be taken not to cause the board to become bowed after installation.

2-11. CARD CAGE. If the 2 inch power supply section is cut off the VAB-2 will fit standard 12" card cages (such as the Mostek XAID development system enclosure).

2-12. VENTILATION. Adequate ventilation must be provided, especially at the power supply (if included). Vertical orientation is preferable for convection cooled systems. External heat sinking may be required in closed systems. For example, a heat conductive standoff may be attached between one of the regulator mounting bolts and the chassis.

2-13. CONNECTIONS.

2-14. POWER.

2-15. AC Operation. For AC operation, 12.6 VAC (nominal) is connected to plated-thru eyelets E20 and E21. Jumpers are installed as follows:

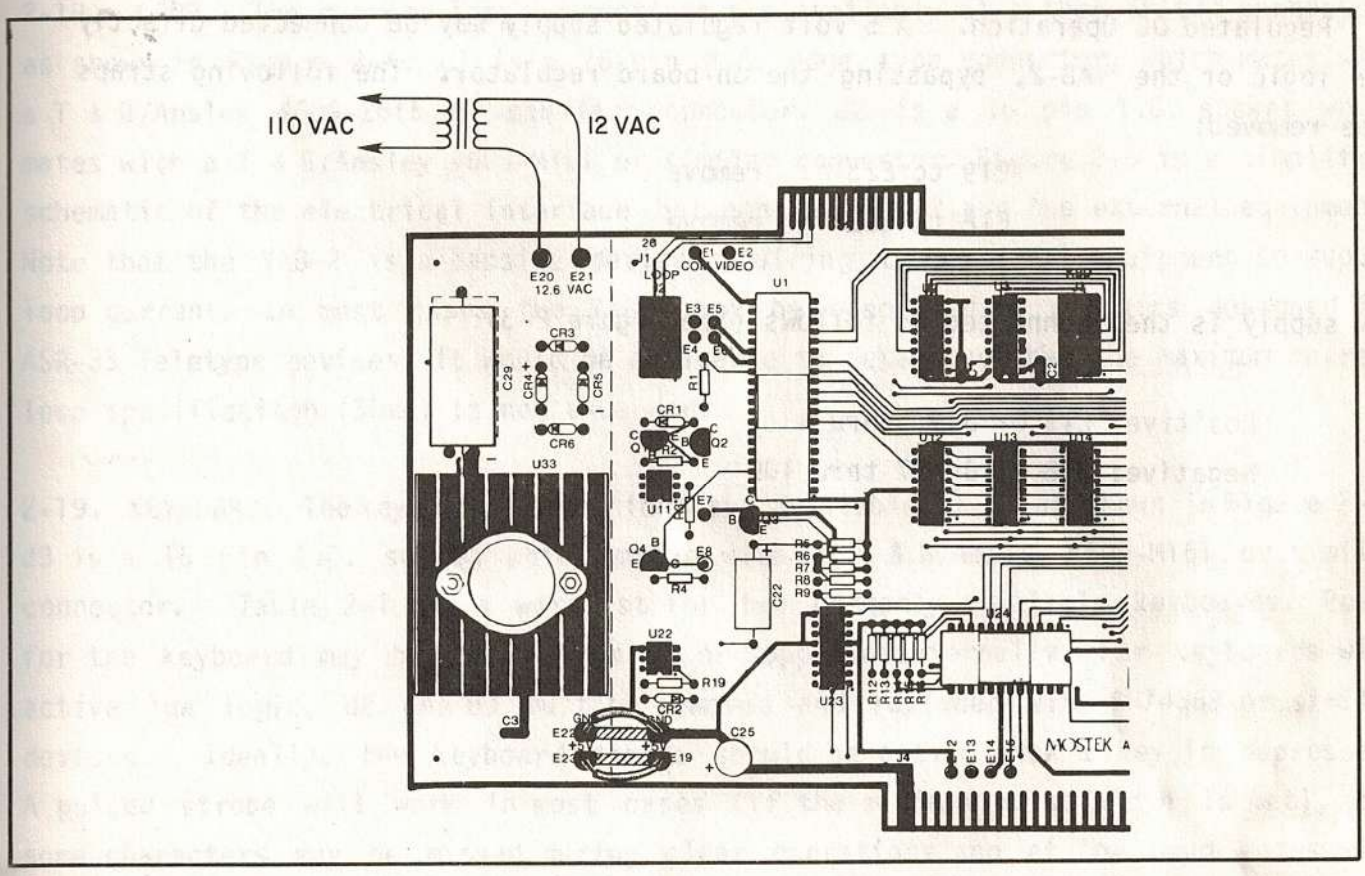
E19 to E23	add
E18 to E22	add

shown in Figure 2-1.

NOTE The DC output of the on board regulator is rated at 5 V $\pm$5% at 1000 mA maximum. The worst case load placed on the regulator by the logic on the VAB-2 is 750 mA maximum. Thus 250 mA of current capacity remains to supply a keyboard (See Figure 2-2). With external DC supplies, keyboards of higher current rating may be used.

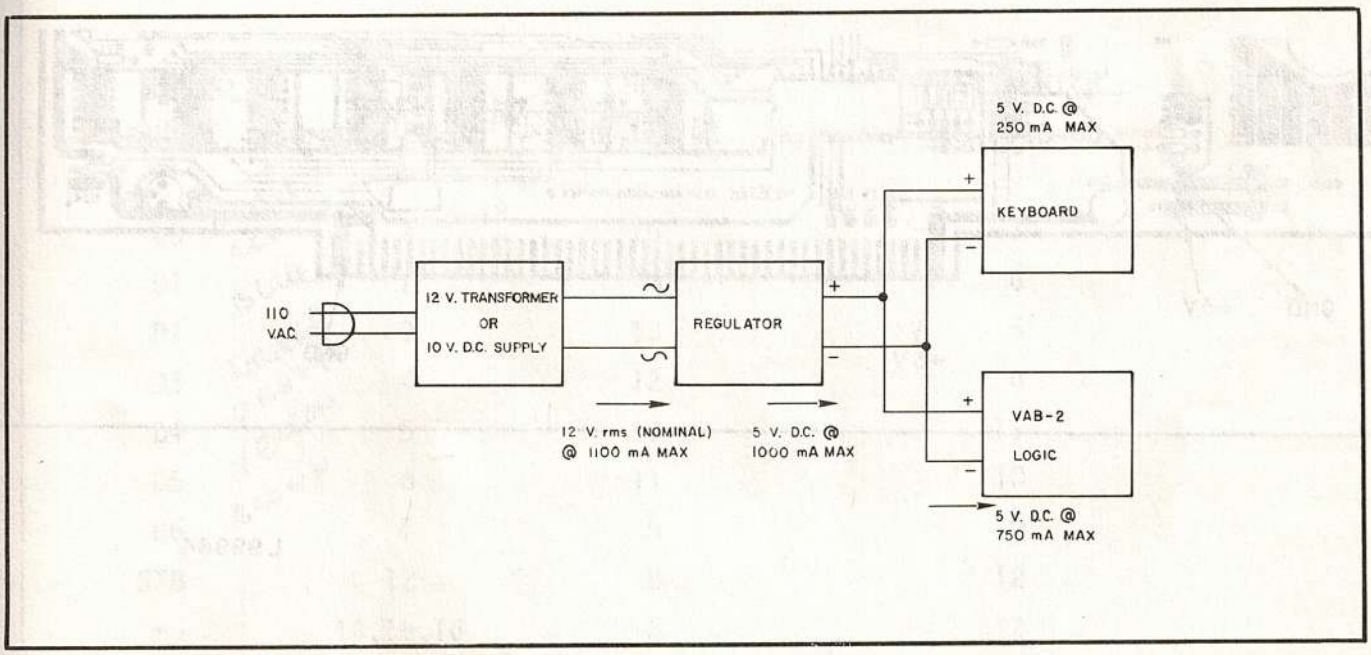
2-16. Unregulated DC Operation. An unregulated DC supply operating between 8 and 10 VDC may be connected in place of the transformer, with hookup and straps as discussed in paragraph 2-15.

Figure 2-1. AC Connections.



L99989

Figure 2-2. DC Power Circuit.



L99987

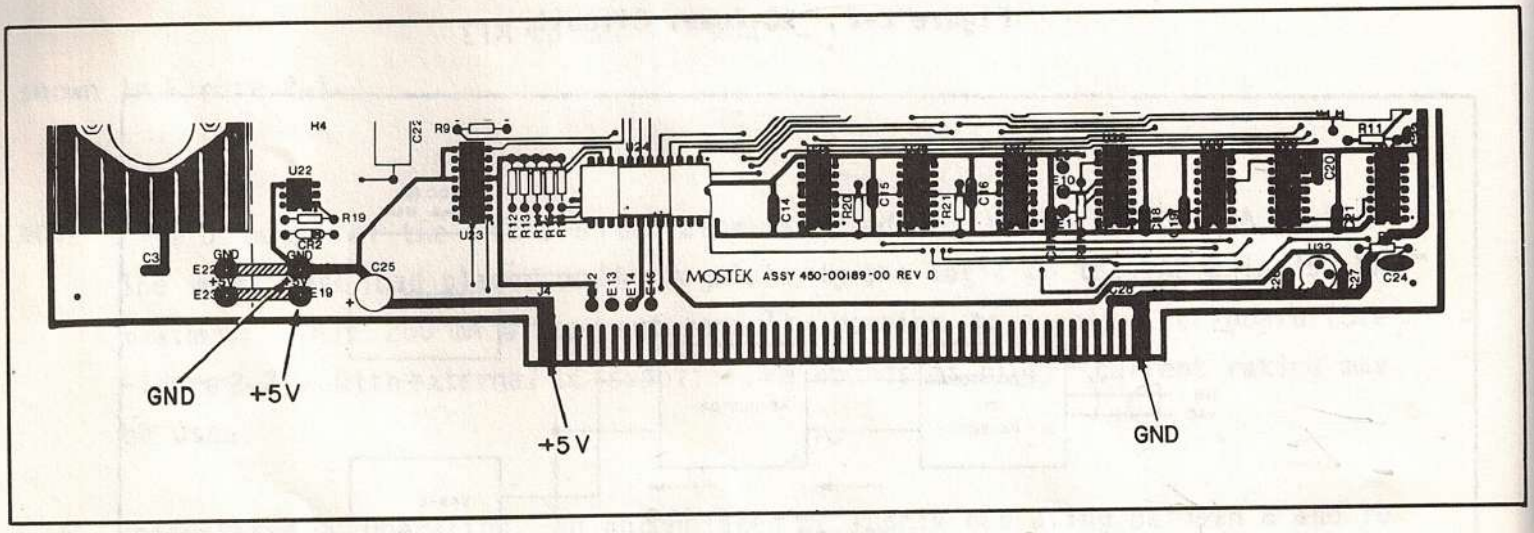
2-17. Regulated DC Operation. A 5 volt regulated supply may be connected directly to the logic of the VAB-2, bypassing the on-board regulator. The following straps must be removed:

E19 to E23	remove
E18 to E22	remove

The 5V supply is then connected as follows (See Figure 2-3):

Positive: E19 or J4-1 thru 4
Negative: E18 or J4-97 thru 100

Figure 2-3. Regulated DC Operation.



L99984

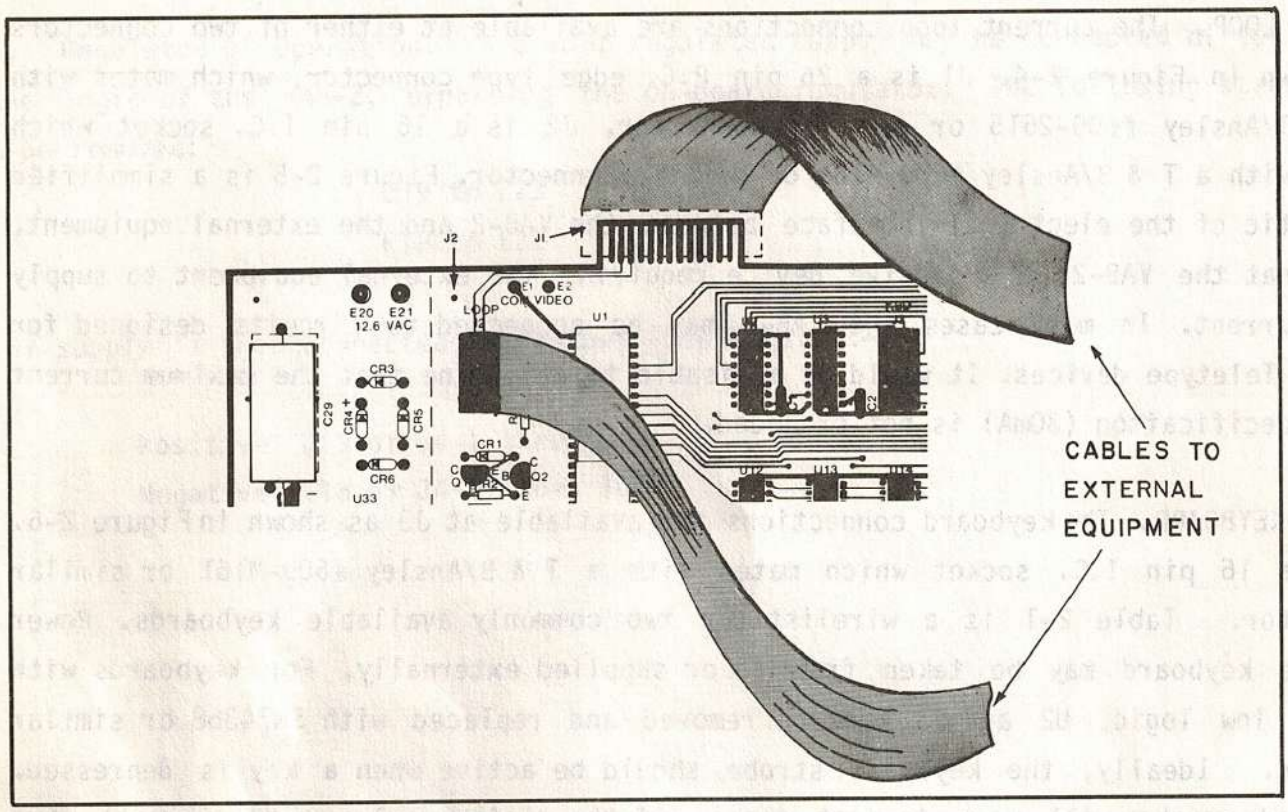
2-18. LOOP. The current loop connections are available at either of two connectors as shown in Figure 2-4. J1 is a 26 pin P.C. edge type connector, which mates with a T & B/Ansley #609-2615 or similar connector. J2 is a 16 pin I.C. socket which mates with a T & B/Ansley #609-M161 or similar connector. Figure 2-5 is a simplified schematic of the electrical interface between the VAB-2 and the external equipment. Note that the VAB-2 is a passive device requiring the external equipment to supply loop current. In most cases the VAB-2 may be connected to circuits designed for ASR-33 Teletype devices. It would be advisable to determine that the maximum current loop specificaiton (30mA) is not exceeded.

2-19. KEYBOARD. The keyboard connections are available at J3 as shown in Figure 2-6. J3 is a 16 pin I.C. socket which mates with a T & B/Ansley #609-M161 or similar connector. Table 2-1 is a wirelist for two commonly available keyboards. Power for the keyboard may be taken from J3 or supplied externally. For keyboards with active low logic, U2 and U3 must be removed and replaced with SN74368 or similar devices. Ideally, the keyboard strobe should be active when a key is depressed. A pulsed strobe will work in most cases (if the minimum pulse width is met), but some characters may be missed during clear operations and at low baud rates with fast typists.

Table 2-1. Keyboard Wirelists.

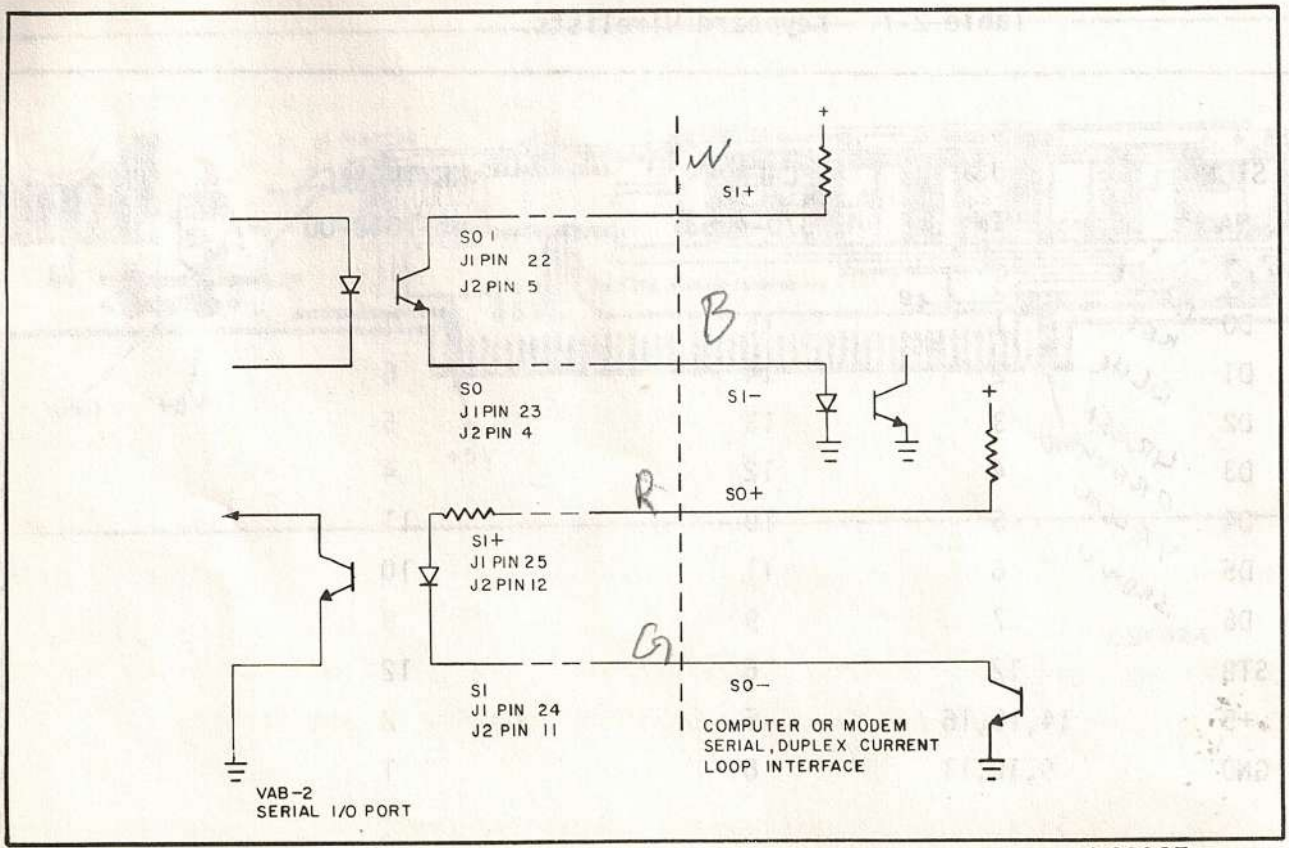
SIGNAL NAME	J3 PIN	CHERRY B70-4753	KEYTRONICS 65-1648-00
D0	1	15	7
D1	2	14	6
D2	3	13	5
D3	4	12	4
D4	5	10	11
D5	6	11	10
D6	7	9	9
STB	12	6	12
+5	14,15,16	5	2
GND	9,10,11	8	1

Figure 2-4. Loop Cables.



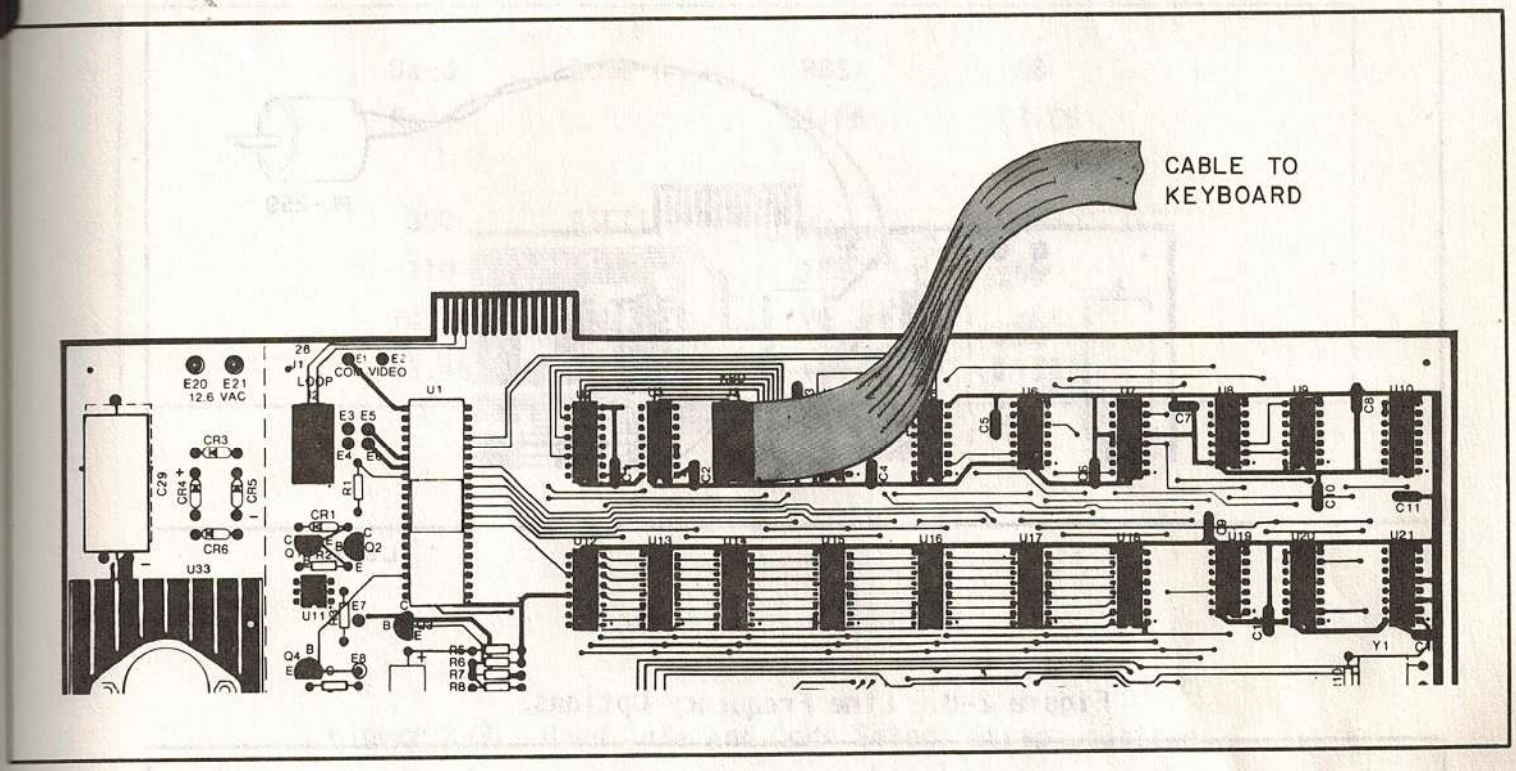
L99984

Figure 2-5.
Simplified Schematic of Loop Connections.



L99983

Figure 2-6. Keyboard Connection.



L99982

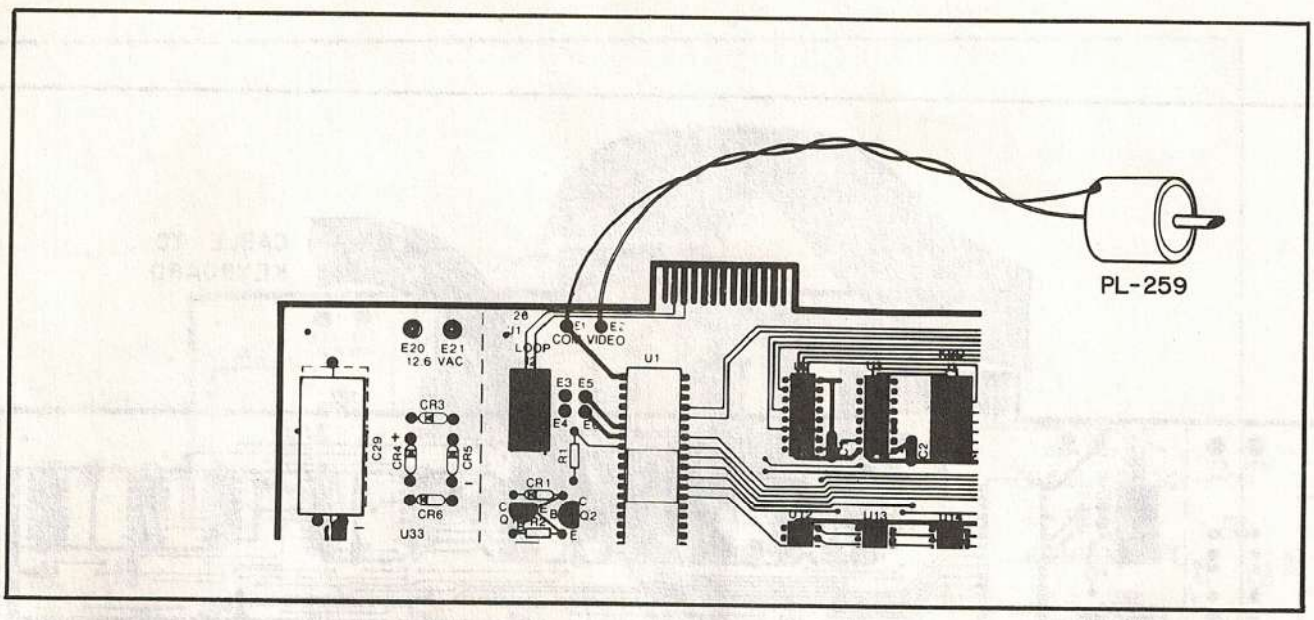
2-20. VIDEO. The VAB-2 is supplied with a twisted pair cable and a PL-259 UHF type connector to mate with most video monitors. (See Figure 2-7)

2-21. STRAP OPTIONS.

2-22. Line Frequency. Figure 2-8 shown the line frequency strap options for 50 and 60 Hz use.

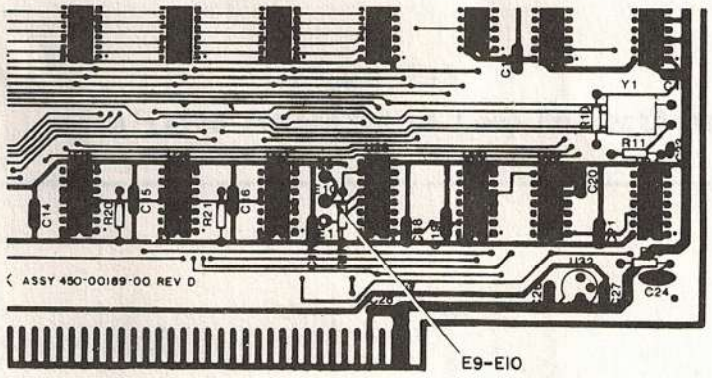
2-23. Baud Rate and Code Select. Table 2-2 gives the strap configurations for the available options. Figure 2-9 shows the physical location of the straps.

Figure 2-7. Video Connection.

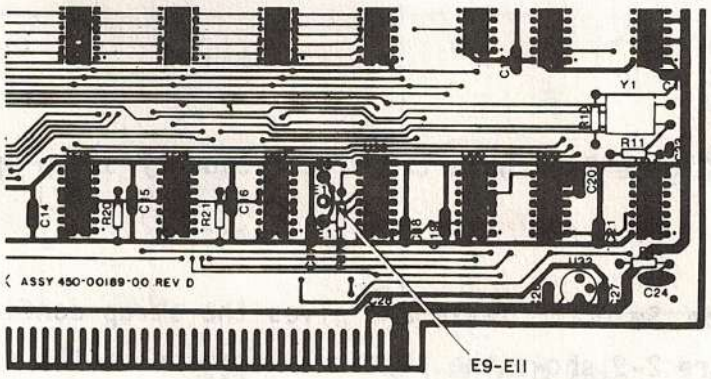


L99981

Figure 2-8. Line Frequency Options.



(a) 50 Hz.



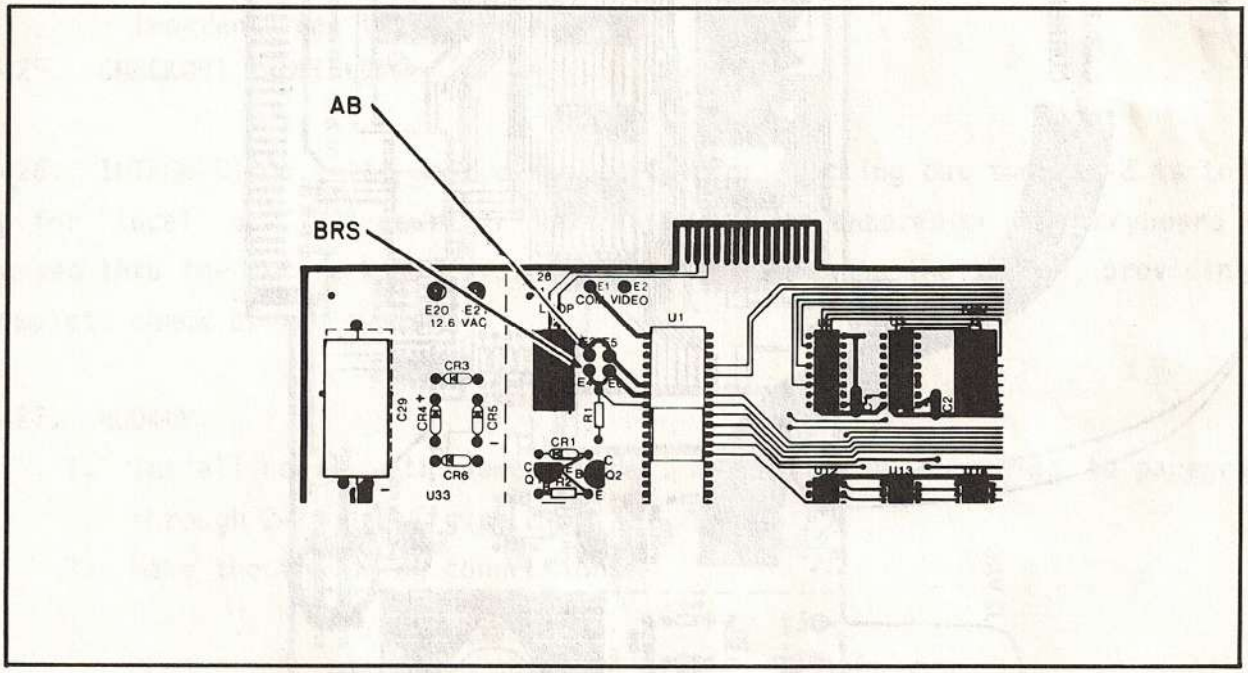
(b) 60 Hz.

L99980

Table 2-2. Baud Rate and Code Select Strap Table.

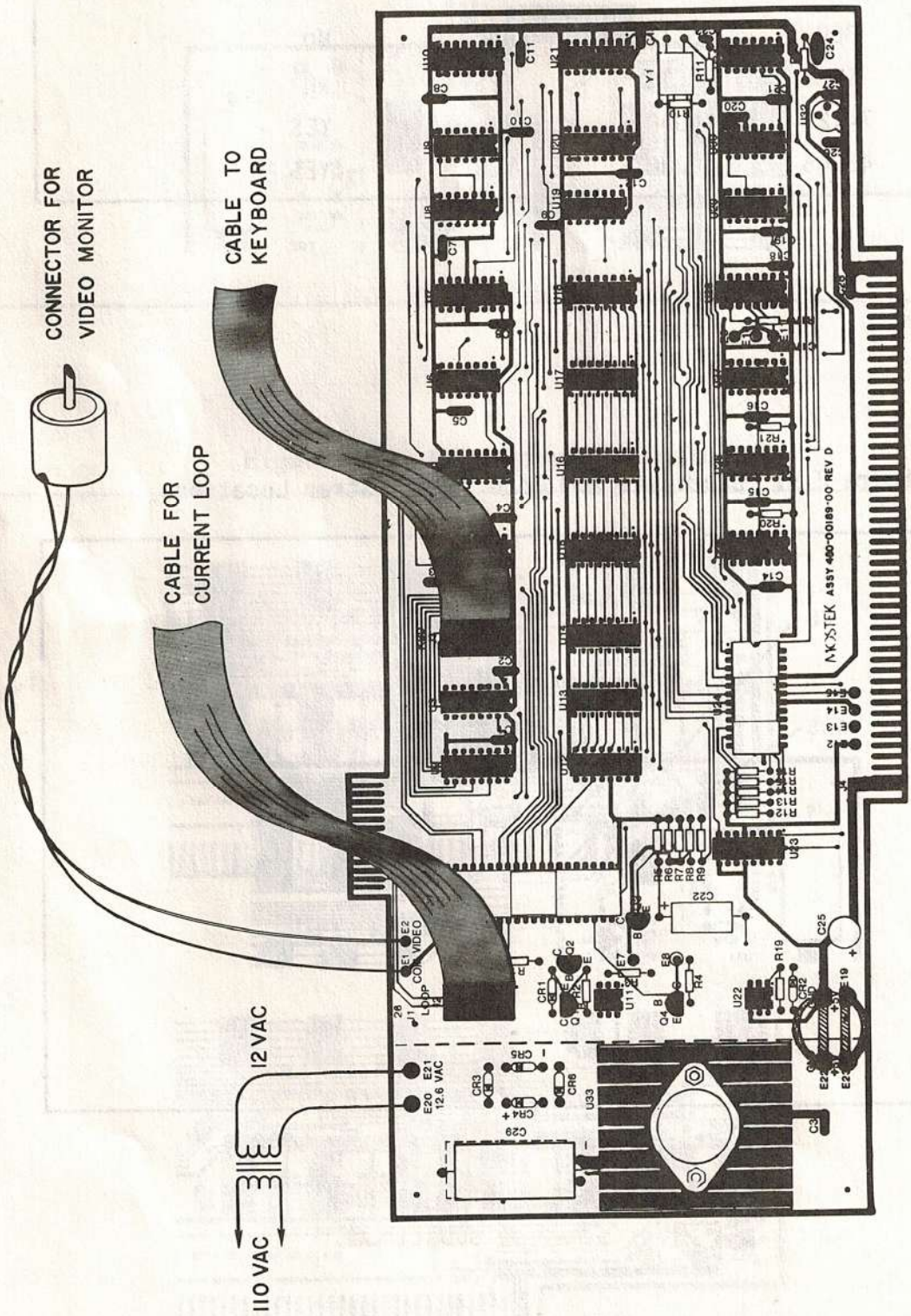
Baud Rate	Code	BRS E4-E6	AB E3-E5
300	ASCII	NO	NO
110	ASCII	YES	NO
74.2	BAUDOT	NO	YES
45.45	BAUDOT	YES	YES

Figure 2-9. Baud Rate and Code Select Strap Locations.



L99979

Figure 2-10. Sample Complete Configuration.



L99978

2-24. Rom Select. Several options are provided for the character generator function. VAB-2 is normally shipped from the factory with an MK34073 character generator ROM, requiring no additional straps. As an option, the user may wish to order and install a custom ROM. A single custom MK34000 series ROM is capable of providing two complete sets of 128 displayable characters and symbols. Pin 19 of the ROM (address A10) is used to select the character set in use. A strap option is provided to facilitate user set up. Table 2-3 details the straps:

Table 2-3. ROM Straps.

A10	low	E15	to	E14	(Note 1)
A10	high	E15	to	E12	(Note 2)

NOTE 1 This strap is present in etch on circuit side of VAB-2 circuit board.

NOTE 2 The etch between E15 and E14 must be removed before installing this strap (warranty may be voided).

2-25. **CHECKOUT PROCEDURE.**

2-26. INTRODUCTION. The most direct method of checking out the VAB-2 is to hook it up for "local" or "TV typewriter" operation. Data entered on the keyboard is then passed thru the entire VAB-2 system before appearing on the screen, providing a very complete check of the system.

2-27. HOOKUP.

1. Install board with power, video, and keyboard according to paragraph 2-13 through 2-24 and Figure 2-10.
2. Make the following connections:

   J2 pin 5 to positive side of DC supply
   J2 pin 4 to J2 pin 12
   J2 pin 11 to negative side of DC supply
   (equivalent J1 pins may be substituted)

2-28. CHECKOUT.

1. Turn on power. Screen should clear, leaving cursor in upper left hand corner.
2. Type and confirm printable characters:

12345678	unshifted
12345678	shifted
ABCDEFGH	unshifted
ABCDEFGH	shifted

3. Type and confirm special printable characters:

control P	nothing happens
NUL	prints α
control P	nothing happens
control A	prints β
control P	nothing happens
line feed	prints ¶

4. Type and confirm control characters:

control L	should clear screen
.	should print dot and leave cursor to right of dot
control H	should move cursor under dot
control I	should move cursor to right of dot
line feed	should move cursor down one line
control K	should move cursor up one line
control K	cursor should not move, but dot should move down one line
control K	dot should move down one line again
carriage return	cursor should move to left margin
ESC	nothing happens
=	nothing happens
?	nothing happens
?	cursor moves to lower right corner of screen
carriage return	cursor moves to lower left corner of screen
ESC	nothing happens
=	nothing happens
NUL	nothing happens
?	cursor moves to upper right corner of screen
control D	cursor moves to upper left corner of screen

"dot return" 15 times	creates a diagonal of dots
control K 8 times	moves cursor up
dot 3 times	prints 3 dots
control H 12 times	positions cursor under dot in diagonal
control E	erases right hand portion of line, including cursor position
M	prints M
control F	erases screen below line with M
control L	clears all of screen

SECTION 3

OPERATION

3-1. INTRODUCTION.

3-2. The VAB-2 consists of two independent functional units: Keyboard transmitter and a monitor screen receiver. The only interaction between the two occurs when certain received control characters initiate time consuming operations on the screen (the clear operations). During those operations the keyboard is not scanned. However, there is no keyboard action which directly causes any action on the screen. Any action to be produced on the screen must be received from the serial loop. Therefore, data transmitted away from the VAB-2 must be echoed back to the VAB-2 by the external equipment if it is deemed desirable to allow keyboard initiated actions. The following sections discuss the operation of the transmitter and receiver as independent units. The user may find it useful to utilize the setup suggested in paragraphs 2-13 through 2-24 as a means of simulating (via the keyboard) data transmitted to the screen.

3-3. RECEIVER OPERATION - ASCII

3-4. DATA FORMAT. The receiver operating in ASCII mode recognizes a 7 bit code with parity bit (which is ignored). A minimum of 1.0 stop bits must be received; longer stop bits are interpreted as idle time.

3-5. PRINTABLE CHARACTERS. Characters in the bottom 3 rows of Figure 3-1 may be displayed on the screen simply by transmitting to the VAB-2 the corresponding 7 bit code. (Parity bit is ignored). To display one of the special printable characters, a two character sequence must be transmitted. The first character of the sequence is a DS (downshift) (020 octal, or 10 hex). The second character is the seven bit code for the displayed character. In effect, the DS character disables the special character decoding function for the character after the DS. Upon reception of the displayable character or character sequence, that character is written to the screen at the current cursor position. Then, unless the cursor is already in the rightmost column, the cursor moves to the right one column. Attempting to move the cursor past the right margin causes the rightmost column to be overwritten with the most recently received character.

Figure 3-1. ASCII Character Set.

3-6. SPECIAL CHARACTERS.

3-7. Cursor Moves. The basic cursor moves available are:

1. HOME - moves cursor to upper left hand corner of screen. Data is not changed.

2. BACKSPACE - moves cursor left one column unless already in left most column. Moving cursor beyond left margin is inhibited. Data is not changed.

3. HORIZONTAL TAB - moves cursor right one column unless already in rightmost column. Moving cursor beyond right margin is inhibited. Data is not changed.

4. LINE FEED - Moves cursor down one line of data. If cursor was already on bottom line of screen, all data on screen is shifted up one line, the previous top line of data is lost, and the new bottom line is cleared. If cursor was not already on bottom line, only the cursor moves, and no data is changed.

5. VERTICAL TAB - moves cursor up one line of data. If cursor was already on top line of screen, all data on screen is shifted down one line previous bottom line of data is lost, and new top line is cleared. If cursor was not already on top line, only cursor moves, and no data is changed.

6. CARRIAGE RETURN - moves cursor to left most column of current line. No data is changed.

3-8. Cursor Sequence. Two special cursor sequences are provided for absolute and relative x-y cursor addressing. Each sequence consists of four characters:

1. ESC (033 octal, or 1B hex)

2. + for relative (053 octal, or 2B hex)
   = for absolute (075 ocata, or 3D hex)

3. vertical address or displacement

4. horizontal address or displacement

Table 3-1. VAB-2 Control Characters

OCTAL	HEX	CTRL	FUNCTION	
004	04	D	HOM	Home - moves cursor to upper left corner of screen
005	05	E	EOL	Erase end of line - erases current line from right margin to current cursor positon (1600 ms max)
006	06	F	EOS	Erase end of screen - erases lines from bottom of screen to, but not including, current line (400 ms max)
010	08	H	BS ←	Back space - moves cursor left one column unless already in left most column
011	09	I	HT →	Horizontal tab - moves cursor right one column unless already in right most column
012	0A	J	LF ↓	Line feed - moves cursor down one line, scrolls screen up if already on bottom line
013	0B	K	VT ↑	Vertical tab - moves cursor up one line, scrolls screen down if already on top line
014	0C	L	FF	Form feed - clears screen and homes cursor (400 ms)
015	0D	M	CR ↵	Carriage return - moves cursor to left margin
020	10	P	DS	Down shift sequence - causes character following DS to be interpreted as printable rather than control. Required for lower 32 symbols (Greek and math), but may be used with any characters.
021	11	Q	DC1	Device control - sets AUX bit to logic 1
023	13	S	DC3	Device control - sets AUX bit to logic 0
033	1B	*	ESC	Start cursor sequence -
				ESC + V H adds V modulo 16 to vertical cursor address
				H modulo 64 to horizontal cursor address
				ESC = V H sets vertical sursor address to V modulo 16
				horizontal cursor address to H modulo 64
177	7F		DEL	Delete - moves cursor left one column, unless cursor was already on left most column; erases new position

*See paragraph 3-22

For relative addressing, the displacement is added to the current cursor valve, and the result is truncated to 4 bits (vertical) or 6 bits (horizontal). As a result, "wrap around" cursor positioning is possible.

3-9. Cursor Examples.

3-10. Cursor is at V=0, H=0 (upper left corner). This sequence is received: "ESC" "+" "A" "C". "A"=41 (hex) which is added to the current V cursor 0 giving 41 (hex). The result is then truncated (modulo 16 decimal) with a result of 1. Similarly, "C" equals 43 (hex), added to H cursor 0 giving 43 (hex). After truncation (modulo 64) the result is 3. Thus the sequence moved the cursor from 0,0 to 1,3.

3-11. Cursor is at V=4, H=5. This sequence is received: "ESC" "+" "?" "?". "?" equals 3F (hex). 3F plus 4 equals 43, which after truncation (mod 16) leaves 3. 3F plus 5 equals 44, which after truncation (mod 64) leaves 4. Thus the sequence moved the cursor from 4,5 to 3,4.

3-12. The intent of the modulo arithmetic is to allow the programmer to specify the address or displacement either directly in binary (convenient for assembly language) or as literal, printable characters (convenient for high level programming). As an example, the following Fortran subroutine will move the cursor to any position, specified in the parameter list as IV, IH.

```
      SUBROUTINE MOVE (IV,IH)
      DIMENSION IH LIST (64)
C
      DATA IO LIST / IH@,ILA,ILB,ILC,...IHY,IHZ,
     C               IH+,IH°,IH-,IH.,IH/,IHO,IHI,...,IH9,
     C               IH:,IH;, IH ,IH=,IH ,IH?/
C
      DATA IESC /27/
C
      WRITE (ICRT,1) IESC, ILIST (IV+1), ILIST (IH+1)
1     FORMAT (1A1,1H=,2A1)
      END
```

(The example assumes that the Fortran compiler utilizes ASCII.)

3-13. Erases. Several erase functions are provided. It should be noted that some of the erase function require more than one character transmission time to complete. After transmission of an erase, one of the following should occur: transmit filler (NULL's) to the VAB-2 as required to delay the proper amount of time, cease transmission for the required amount of time, or expect to lose some number of characters immediately after the erase. In addition, the keyboard is not scanned during erases.

1. FORM FEED (FF) - FF clears the entire screen and leaves the cursor in the upper left corner (home position). 400 mS is required. (12 characters at 300 Baud, 4 characters at 110 Baud).

2. ERASE TO END OF SCREEN (EOS) - EOS erases complete lines, beginning with the bottom line, and continuing up to but not including the line containing the current position of the cursor. If the cursor is already on the bottom line, no erase occurs. 400 mS maximum required (12 characters at 300 Baud, 4 at 110) for full screen; proportionally less for fewer lines.

3. ERASE TO END OF LINE (EOL) - EOL erases characters on current line, beginning at right margin and moving left upto and including original cursor position. 1600 mS maximum required (48 characters at 300 Baud, 16 at 110) for full line; proportionally less for fewer spaces.

4. DELETE - Delete moves the cursor to the left one position (but not past left margin) then erases the character at the new position. Delete is functionally equivalent to Backspace Space Backspace.

3-14. Device Control. A special output signal, designated AUX, is available at J4 pin 26 for custom applications. The AUX pin is set to logic 1 at power up and upon receipt of a DC3 character (023 octal, or 13 hex). The AUX pin is set to logic 0 upon receipt of a DC1 character (021 octal, or 11 hex). The state change will occur within 1 ms after receipt of the device control character. Possible applications for the AUX pin include:

1. peripheral control
    a. cassette motor stop start
    b. printer enable/disable

2. character set select (wired to E15, see Section 2-20)

3. status indication (buffered, driving LED or other lamp)

3-15. RECEIVER OPERATION - BAUDOT.

3-16. THE BAUDOT CODE. The Baudot code was invented for use with early mechanical teleprinter systems. A 5 bit code is used, giving 32 possible binary combinations. Since the alphabet and number system requires 36 characters, and punctuation was also to be included, a provision was made to give most of the 32 binary values more than one meaning, depending upon retention of previously transmitted information. The Baudot teleprinter has a type basket similar to common typewriters. The type basket has two mechanical positions, one causing letters to be printed, the other causing figures and punctuation. Two binary codes were selected as shift characters: "Letters" (abbreviated LETS) causes the basket to lock into letters position; "Figures" (FIGS) causes the basket to lock into figures and punctuation. Figure 3-2 shows the Baudot character and the relationship between LETS and FIGS cases. The VAB-2 retains a "case history", enabling the proper display of letters, figures and punctuation.

3-17. DATA FORMAT. The receiver operating in Baudot mode utilizes a 5 bit code with no parity bits. A minimum of 1.0 stop bits must be received; longer stop bits are interpreted as idle time.

3-18. PRINTABLE CHARACTERS. Figure 3-2 shows the Baudot character set (in alphabetical order for ease in locating corresponding letter - figure combinations).

FIGURE 3-2. BAUDOT CHARACTER SET.

LETS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
FIGS	-	?	:	*	3	$	&	#	8		(	)	.	,	9	0	1	4	!	5	7	;	2	/	6	"

3-19. CONTROL CHARACTERS. Only 5 control characters are used in Baudot mode:

1. CARRIAGE RETURN - moves cursor to left margin and shifts to "letters" case.

2. LINE FEED - moves cursor down one line (Refer to paragraph 3-1 for details), and shifts to "letters" case.

3. SPACE - "prints" a blank and moves cursor to the right, then shifts to "letters" case

4. LETS - causes no action on screen, shifts to "letters" case

5. FIGS - causes no action on screen, shifts to "figures" case

3-20. **TRANSMITTER OPERATION - ASCII.**

3-21. DATA FORMAT. The transmitter in ASCII mode transmits a 7 bit code with parity bit (set to "0" or spacing) and 2.0 stop bits. The distant end receiver, if looking for less than 2.0 stop bit, will interpret the extra bits as idle time.

3-22. KEYBOARD OPERATIONS. ASCII encoded keyboards vary so greatly from manufacturer to manufacturer that it is difficult to establish any general guidelines for keyboard operations. The following comments are a compendium of keyboard characteristics. The user must determine which apply to his particular keyboard.

1. UPPER CASE LETTERS - On upper-case-only keyboards, depress only the appropriate letter key. On upper-lower case keyboards, depress and hold "shift", then depress appropriate letter key.

2. LOWER CASE LETTERS - On upper-lower case keyboards, depress only the appropriate letter key. On upper-case-only keyboards, lower case letters are not generally produceable.

3. TTY MODE - On some upper-lower-case keyboards, depressing the TTY mode key locks the letter keys into upper-case-only, but generally has little or no effect on other keys.

4. NUMBERS - Depress the appropriate number key.

5. PUNCTUATION AND SYMBOLS - Generation of these symbols are highly keyboard

dependent. See Figure 3-3 for some typical key top layouts. Symbols in Zone 1 require only depression of that key. Symbols in zone 2 are generated by depressing and holding "shift" and then depressing the symbol key.

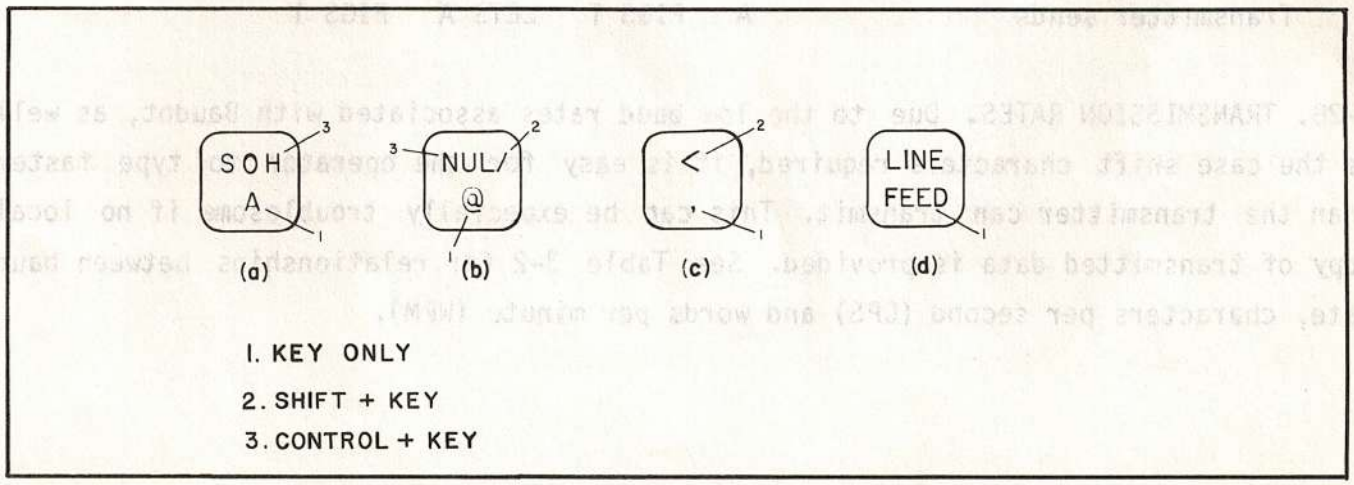
6. CONTROL CHARACTERS - Keys dedicated to control functions (See example in Figure 3-3d) require only depression of that key. Typically carriage return, line feed, "rubout" or "delete", and occasionally "ESC" are provided dedicated keys. Most control codes (including those provided with dedicated keys) may also be generated using the control key. As shown in Figure 3-3 (a) and (b), many codes require that the "control" key be depressed and held, and the appropriate code key be depressed. On upper-case-only keyboards, typically 26 control codes may be generated from:

	CNTL A,	CNTL B,	...,	CNTL Y,	CNTL Z
Hex	01	02		19	1A

On upper-case-only keyboards, the remaining six codes may often be generated by simultaneously depressing and holding "control" and "shift" and then depressing K thru P:

	c/s P	, c/s K	, c/s L	, c/s M	, c/s N	, c/s O
Hex	00	1B	1C	1D	1E	1F

Figure 3-3. Typical Keytops.



L 99977

The combined control-shift mechanism does not always work, especially on upper-lower case keyboards. On such keyboards, NUL (hex 00) and ESC (hex 1B) often have dedicated keys, and the other codes (hex 1C thru 1F) are produceable by depressing control and a symbol key.

3-23. **TRANSMITTER OPERATION - BAUDOT.**

3-24. DATA FORMAT. The transmitter in Baudot mode transmits a 5 bit code with no parity and 2.0 stop bits. The distant end receiver, if looking for less than 2.0 stop bits, will interpret the extra bits as idle time.

3-25. KEYBOARD OPERATIONS.

3-26. Control Characters. The small number of control characters useable in Baudot mode are generally available as dedicated keys on ASCII keyboards. LETS and FIGS, respectively are produced by ASCII keys rubout (delete) and NULL.

3-27. Printable Characters. Only a small subset of the ASCII character set are implemented in the Baudot code (See Figure 3-2). Those characters are usually generated by simply depressing a single key, or at most, "shift" and one other key. ASCII characters entered on the keyboard that have no Baudot equivalent are ignored. The transmitter retains a history of the case ("LETS" or "FIGS") of the most recently transmitted character. Appropriate shift characters are automatically generated by the transmitter as required. For example:

Operator enters	A	1		A		1	
Transmitter sends	A	FIGS	1	LETS	A	FIGS	1

3-28. TRANSMISSION RATES. Due to the low baud rates associated with Baudot, as well as the case shift characters required, it is easy for the operator to type faster than the transmitter can transmit. This can be expecially troublesome if no local copy of transmitted data is provided. See Table 3-2 for relationships between baud rate, characters per second (CPS) and words per minute (WPM).

Table 3-2. Transmission Rates.

Baud	CPS	WPM	
45.45	6	60	(1)
74.2	10	100	(1)
110.	10	100	(2)
300.	30	300	(2)

(1) May be less if frequent case shifts are required

(2) ASCII rates included for reference

SECTION 4

THEORY OF OPERATION

4-1. INTRODUCTION.

4-2. The VAB-2 consists of:

1. one MK3870 single chip microcomputer
2. seven MK4102 static RAM's for screen memory
3. one MK34073 ROM for character generation
4. TTL random logic for timing and interface
5. one power supply

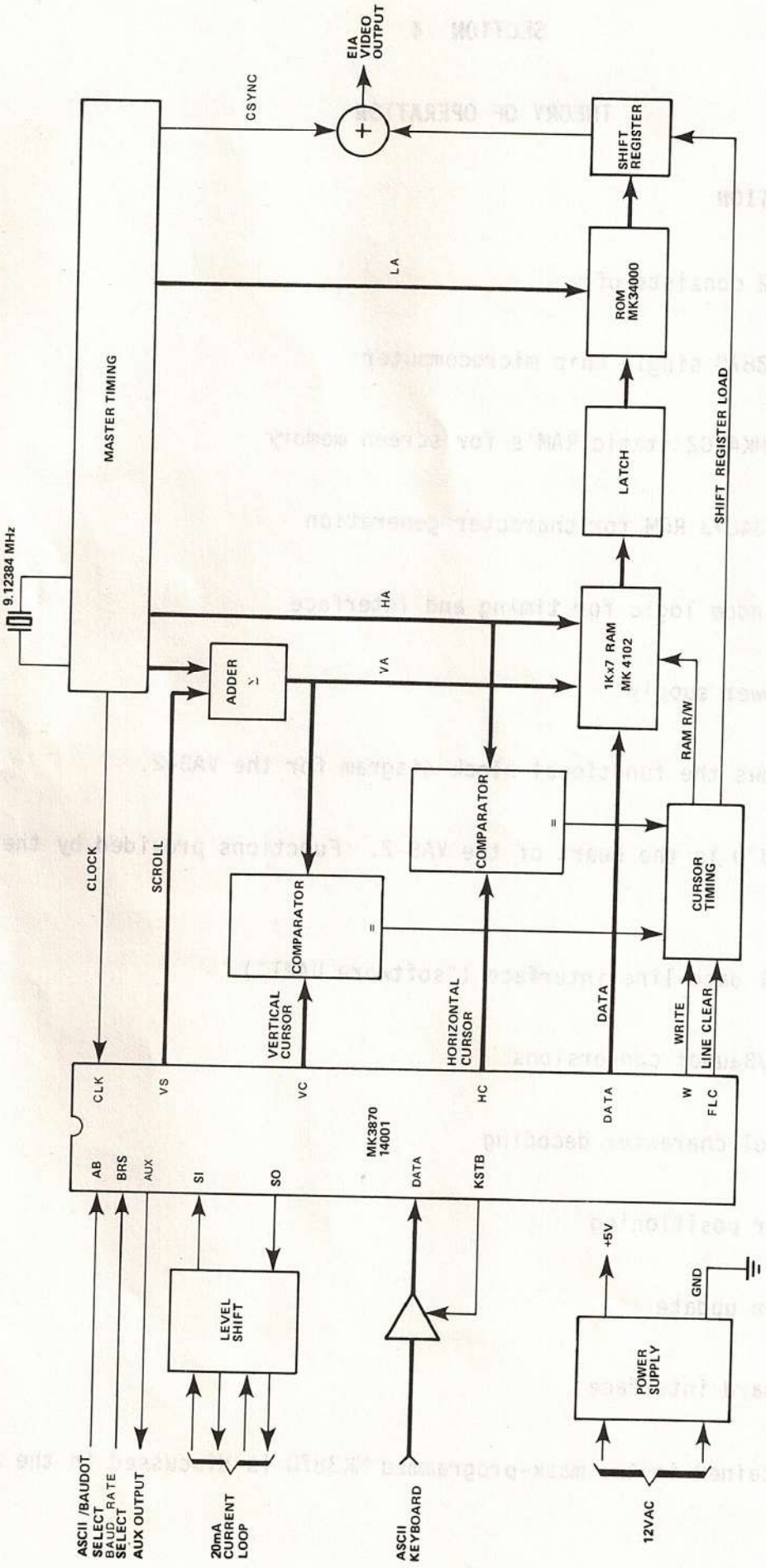
Figure 4-1 shows the functional block diagram for the VAB-2.

4-3. The MK3870 is the heart of the VAB-2. Functions provided by the MK3870 include:

1. serial data line interface ("software UART")
2. ASCII/Baudot conversions
3. control character decoding
4. cursor positioning
5. screen update
6. keyboard interface

Software contained in the mask-programmed MK3870 is discussed in the Appendix.

Figure 4-1. Functional Block Diagram.



L 99976

4-4. BASIC TIMING.

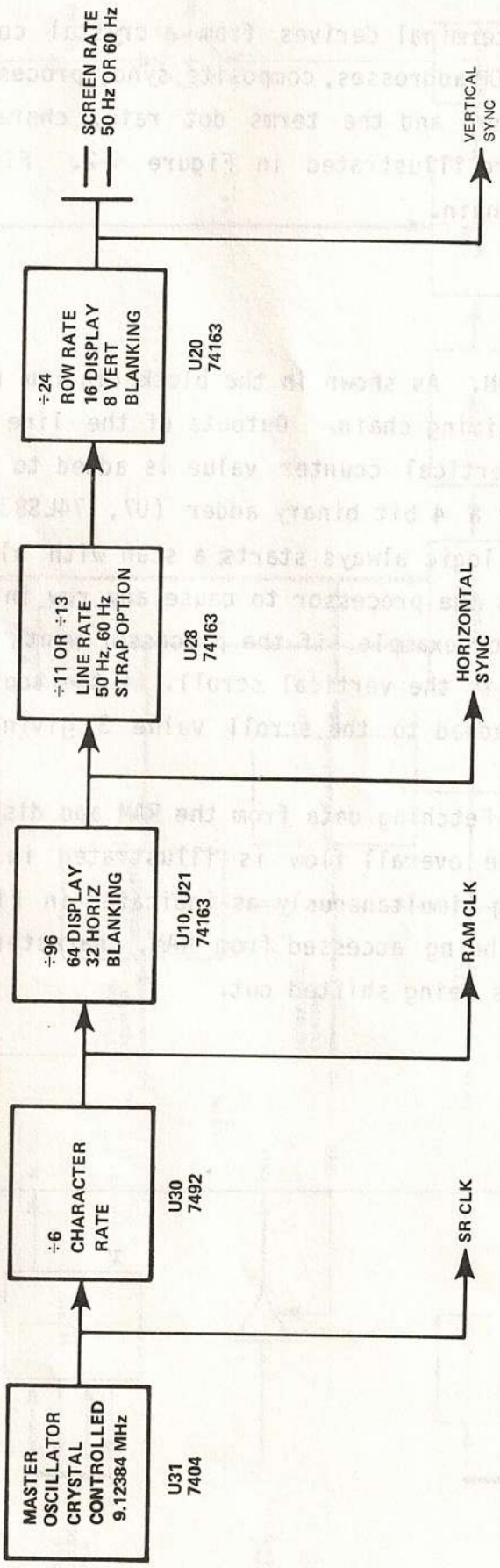
4-5. Timing for the terminal derives from a crystal controlled TTL divider chain that generates RAM and ROM addresses, composite sync, processor clock and other control signals. This division, and the terms dot rate, character rate, line rate, row rate and screen rate are illustrated in Figure 4-2. Figure 4-3 shows the actual outputs of the timing chain.

4-6. VIDEO REFRESH.

4-7. ADDRESS GENERATION. As shown in the block diagram (Figure 4-1), RAM addresses are derived from the timing chain. Outputs of the line and horizontal counter are used directly. The vertical counter value is added to the vertical scroll value (from the processor) by a 4 bit binary adder (U7, 74LS83) to generate the vertical address. The refresh logic always starts a scan with all addresses equal to zero. The scroll adder allows the processor to cause any row in RAM to be displayed at the top of the screen. For example, if the processor wants row 3 to be at the top of screen, it outputs a 3 to the vertical scroll. At the top of the screen the vertical counter value zero is added to the scroll value 3 giving vertical address 3, etc.

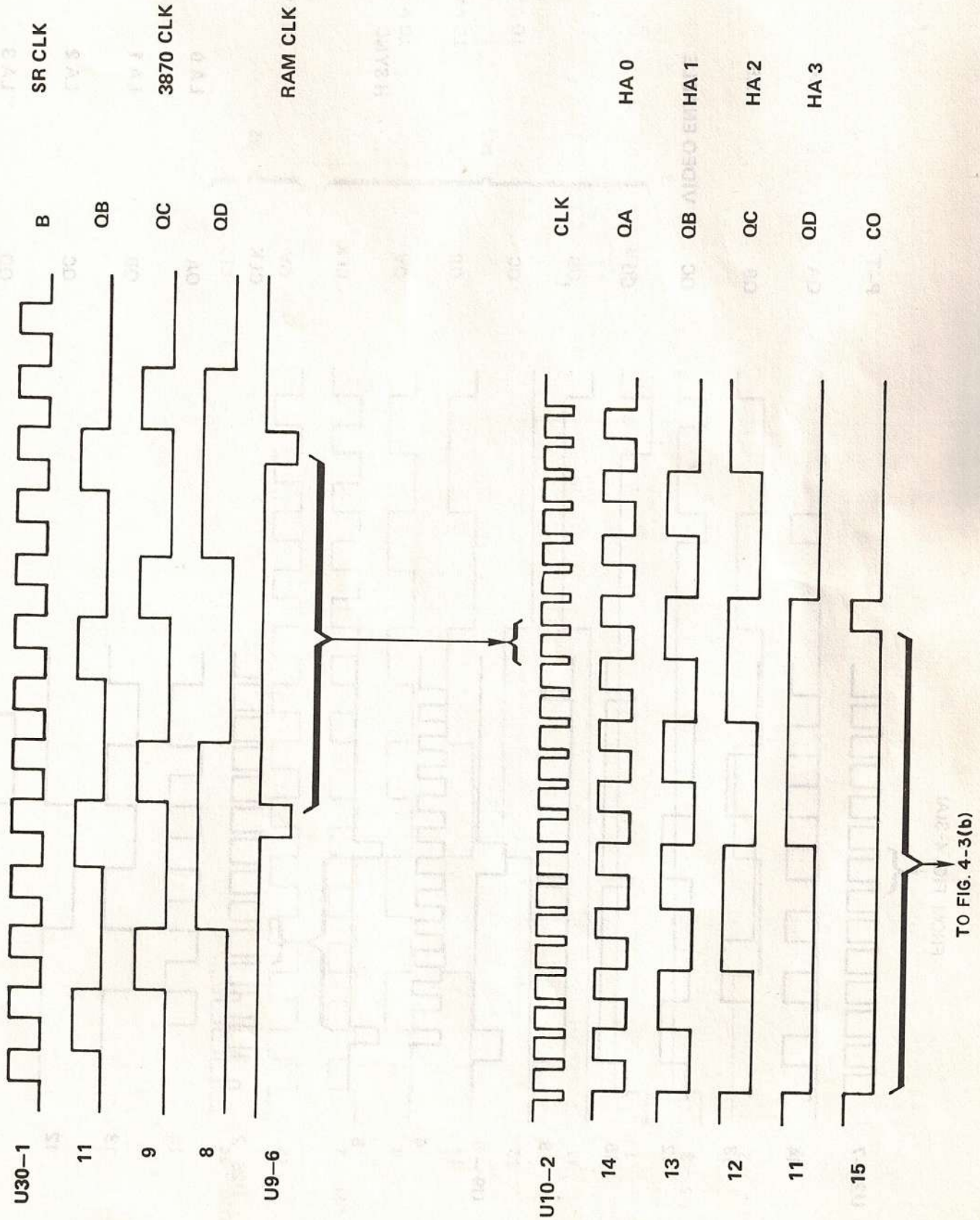
4-8. DATA RETRIEVAL. Fetching data from the RAM and displaying it on the screen is a serial process. The overall flow is illustrated in Figure 4-4 (a). All three processes are occurring simultaneously as indicated in Figure 4-4 (b). For example, while character 3 is being accessed from RAM, character 2 is being accessed from ROM, and character 1 is being shifted out.

Figure 4-2. Video Timing Block Diagram.



L99975

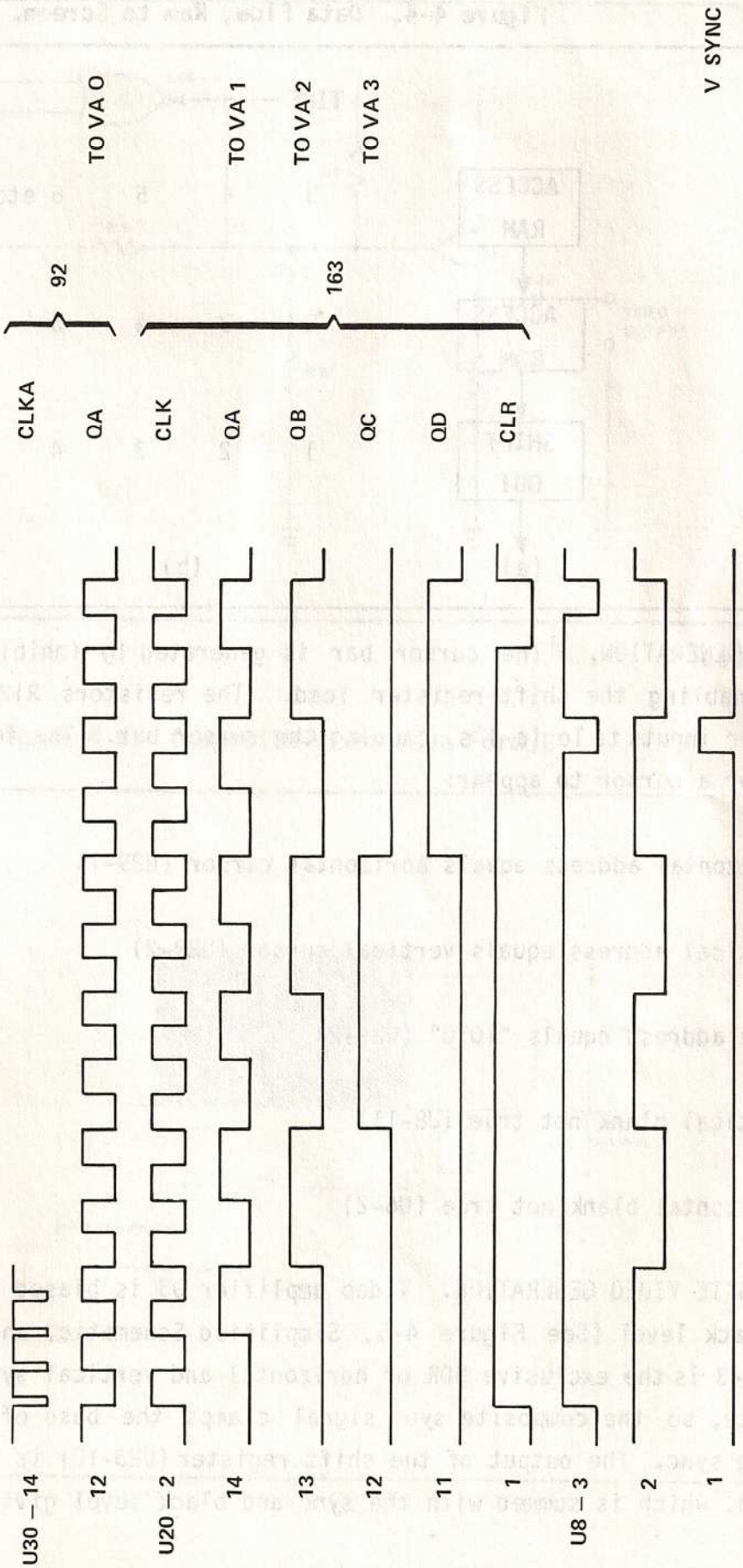
Figure 4-3a. VAB-2 Timing Diagram.



L99974

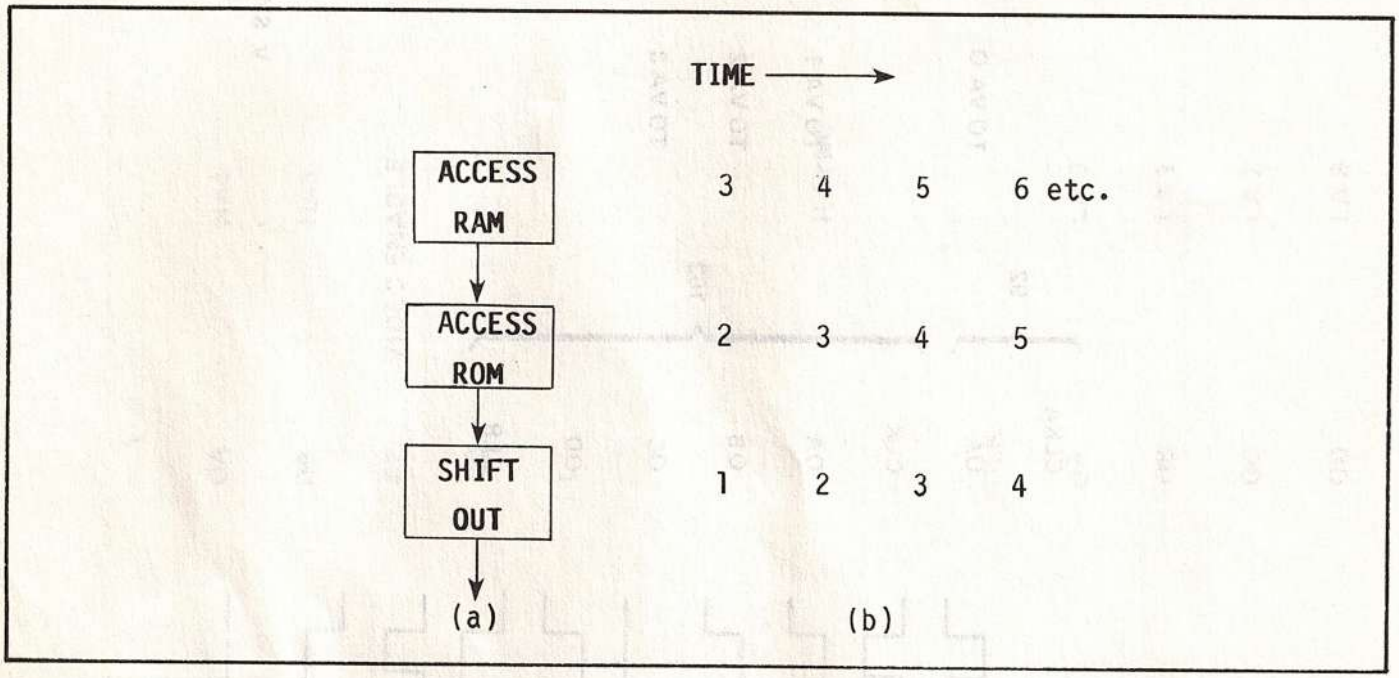
Figure 4-3b. VAB-2 Timing Diagram.

Figure 4-3c. VAB-2 Timing Diagram.



L99972

Figure 4-4. Data Flow, Ram to Screen.



4-9. CURSOR GENERATION. The cursor bar is generated by inhibiting the ROM chip select and enabling the shift register load. The resistors R12 thru R16 pull the shift register inputs to logic 1's, causing the cursor bar. The following conditions must occur for a cursor to appear:

1. horizontal address equals horizontal cursor (U29-1)

2. vertical address equals vertical cursor (U29-2)

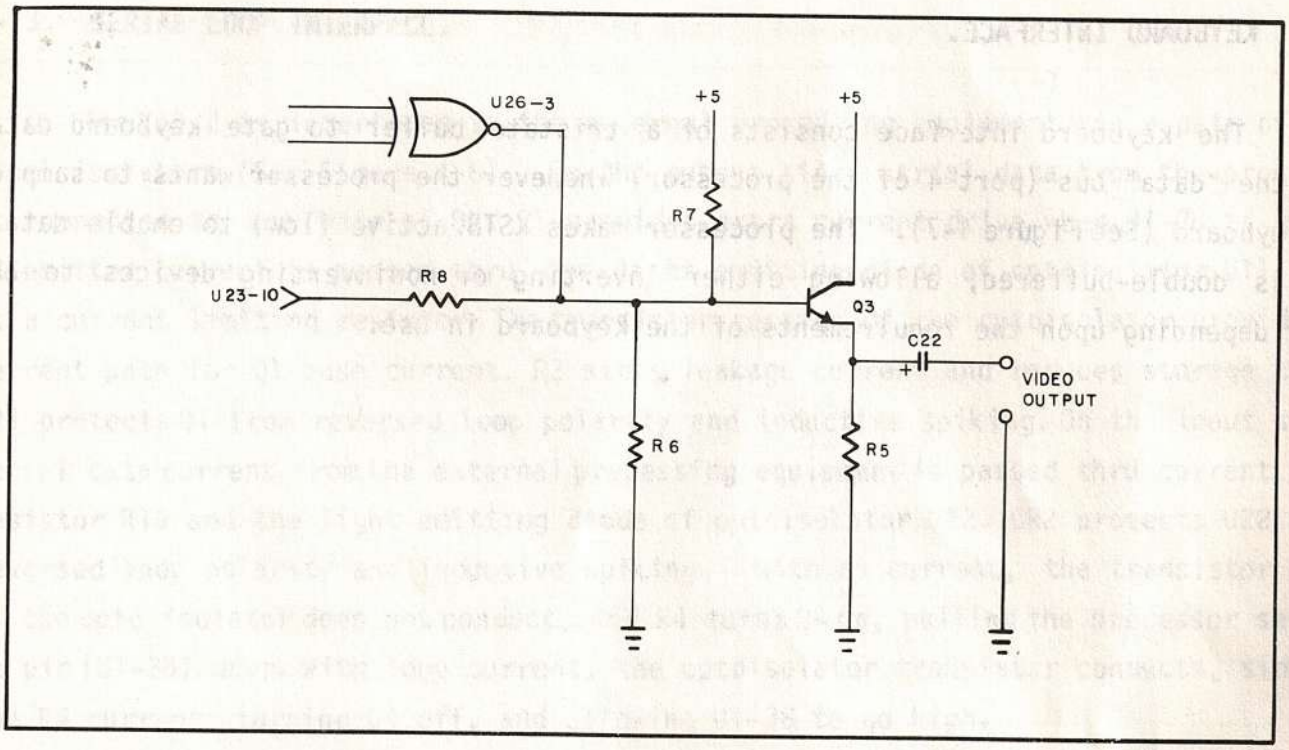
3. line address equals "1010" (U29-2)

4. vertical blank not true (U8-11)

5. horizontal blank not true (U6-2)

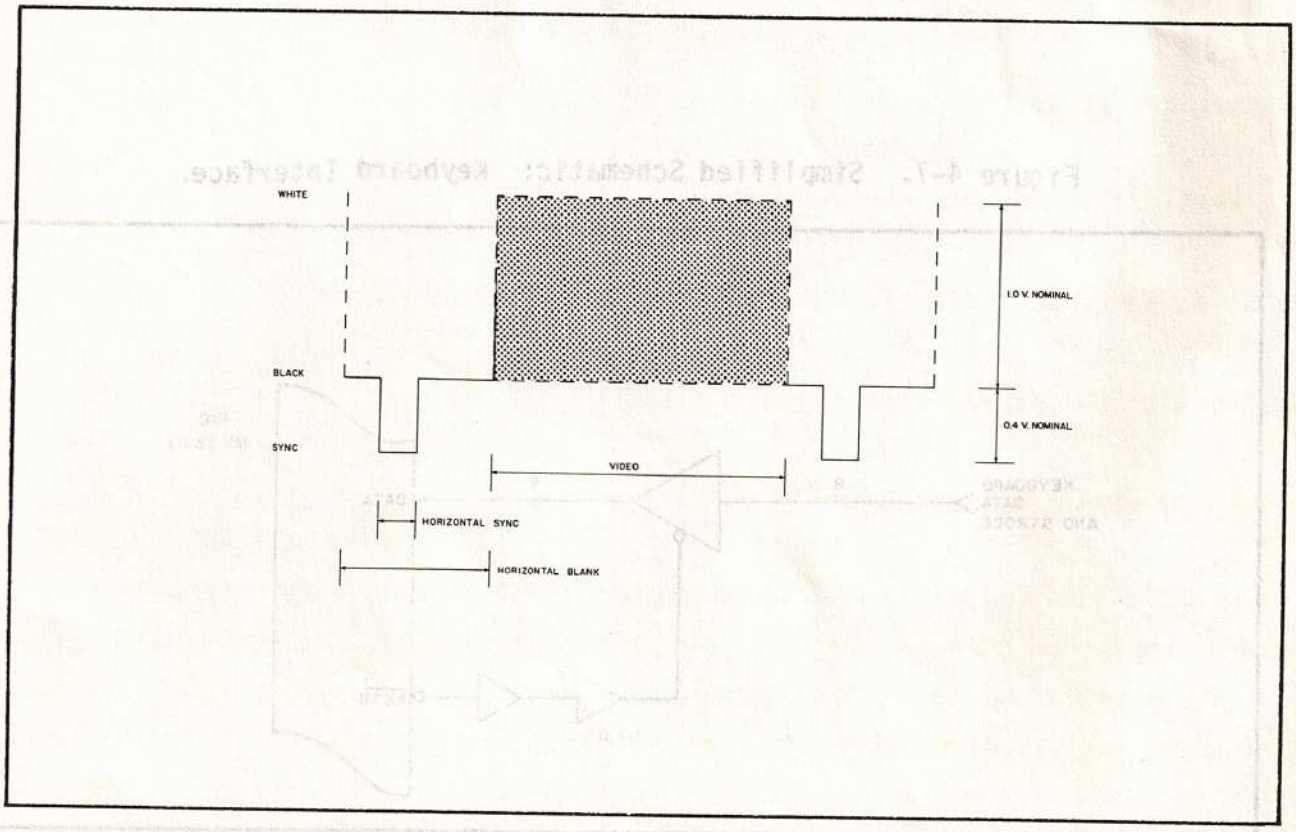
4-10. COMPOSITE VIDEO GENERATION. Video amplifier Q3 is biased by R6 and R7 to the reference black level (See Figure 4-5, Simplified Schematic, and Figure 4-6, Video Signal). U26-3 is the exclusive NOR of horizontal and vertical sync. U26 is an open collector gate, so the composite sync signal clamps the base of Q3 to ground (sync level) during sync. The output of the shift register (U23-10) is applied to the base of Q3 thru R8, which is summed with the sync and black level giving composite video.

Figure 4-5. Simplified Schematic: Video Amplifier.



L99971

Figure 4-6. Composite Video Wave Form.

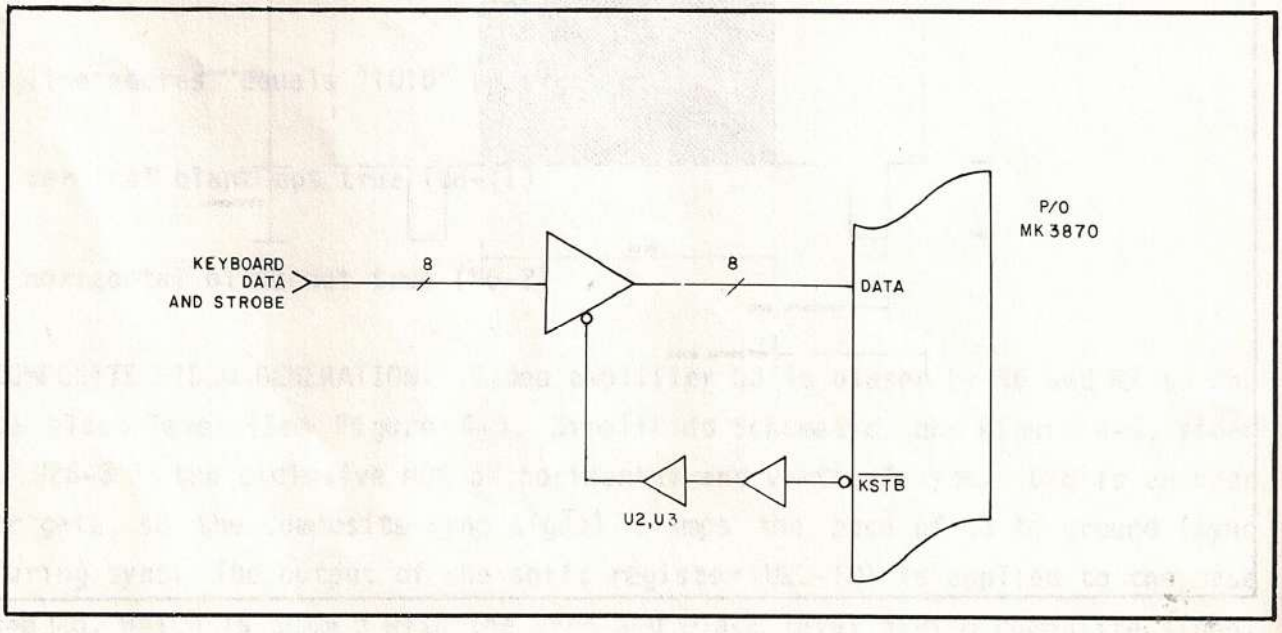


L99969

4-11. KEYBOARD INTERFACE.

4-12. The keyboard interface consists of a tristate buffer to gate keyboard data onto the "data" bus (port 4 of the processor) whenever the processor wants to sample the keyboard (See Figure 4-7). The processor makes KSTB active (low) to enable data. KSTB is double-buffered, allowing either inverting or noninverting devices to be used, depending upon the requirements of the keyboard in use.

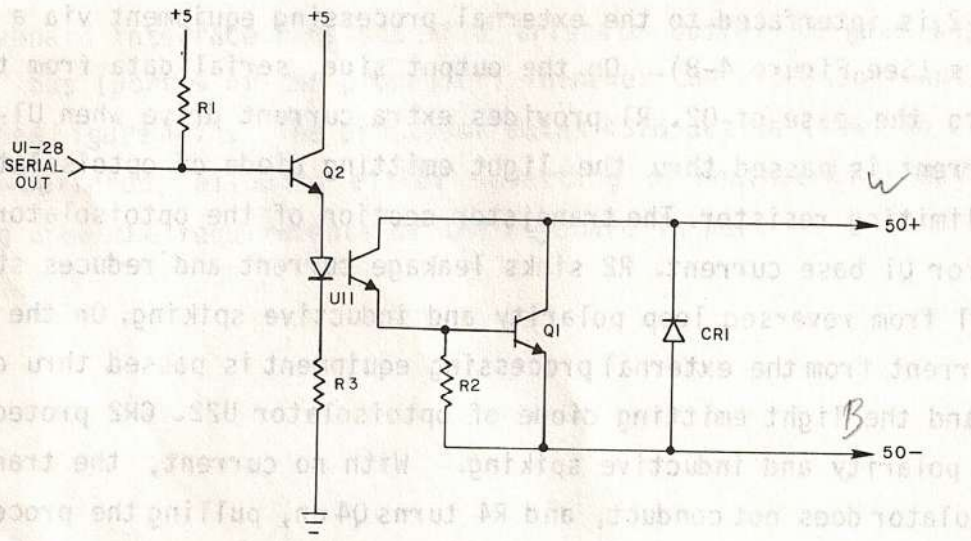
Figure 4-7. Simplified Schematic: Keyboard Interface.



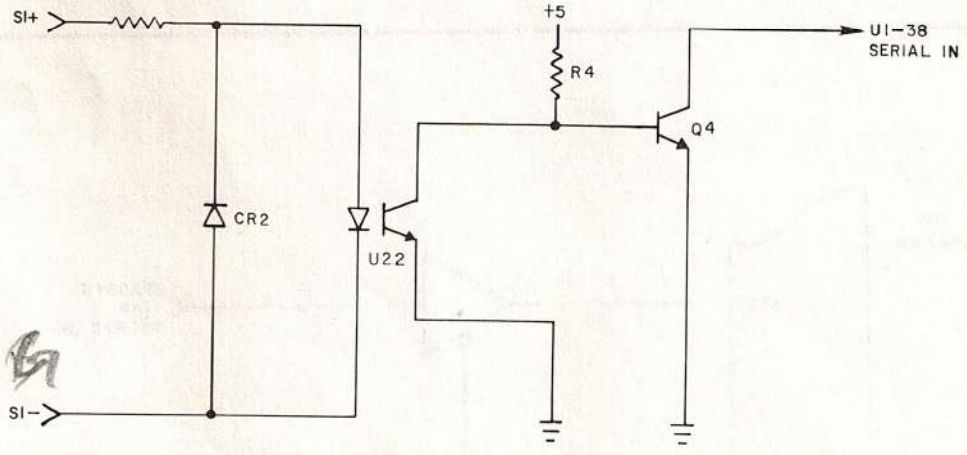
4-13. SERIAL LOOP INTERFACE.

4-14. The VAB-2 is interfaced to the external processing equipment via a pair of optical isolators (See Figure 4-8). On the output side, serial data from the processor is connected to the base of Q2. R1 provides extra current drive when U1-28 is high. Q2 emitter current is passed thru the light emitting diode of optoisolator U11. R3 is a current limiting resistor. The transistor section of the optoisolator provides a current path for Q1 base current. R2 sinks leakage current and reduces storage time. CR1 protects Q1 from reversed loop polarity and inductive spiking. On the input side, serial data current from the external processing equipment is passed thru current loop resistor R19 and the light emitting diode of optoisolator U22. CR2 protects U22 from reversed loop polarity and inductive spiking. With no current, the transistor side of the opto isolator does not conduct, and R4 turns Q4 on, pulling the processor serial in pin (U1-38) down. With loop current, the optoisolator transistor conducts, sinking the R4 current, turning Q4 off, and allowing U1-38 to go high.

Figure 4-8. Simplified Schematic: Loop Interface.



(a)



(b)

L99968

SECTION 5

MAINTENANCE

5-1. PREVENTIVE MAINTENANCE.

5-2. No preventive maintenance is required.

5-3. TROUBLESHOOTING.

5-4. Troubleshooting may be simplified by following the hookup and checkout procedures of Paragraphs 2-25 through 2-28. If the primary symptom is no response to keyboard, it will greatly speed up troubleshooting to first determine if it is the sending or the receiving side that is faulty. The quickest procedure is to repetitively depress the keyboard and check U1-28 and U1-38 for pulse trains. Table 5-1 will help localize the trouble:

Table 5-1. Simplified Troubleshooting.

U1-28	U1-38	LOOK HERE
NO	NO	keyboard circuit
YES	NO	loop circuit
YES	YES	video circuit

Table 5-2 is a list of common failures, their causes, and suggested procedure to isolate the fault.

Table 5-2. Common Failures.

SYMPTOM	POSSIBLE CAUSE	CORRECTIVE ACTION
1. No sync, no cursor	1a. power not applied to VAB-2	
	1b. video cable not connected	
	1c. monitor intensity set too low	
	1d. video amplifier (Q3) defective	1d. check Q3 base
	1e. VAB-2 crystal oscillator not working	1e. check U31-12
2. Sync, no cursor	2a. U23 defective	*
3. Screen does not clear on power up, cursor in upper left corner	3a. U12 thru U18 defective (one or more devices)	*
	3b. U4 or U5 defective	*
	3c. U24 defective or missing	*
	3d. power to U24 prom not applied	
	3e. U23 defective	*
4. Screen does not clear on power up, cursor not in upper left corner	4a. U1 missing or defective	4a. replace
	4b. U1-39 grounded	4b. remove ground
	4c. no clock signal to U1-2	*
5. No response to keyboard	5a. keyboard not connected	
	5b. power not applied to keyboard	
	5c. U2 or U3 defective or missing	*

* Factory service recommended

NOTE Lack of response to keyboard may also be due to data not being transferred to screen. It is therefore necessary to determine that data is being transmitted by the VAB-2 before faults may be pinpointed.

Table 5-2. Cont'd.

SYMPTOM	POSSIBLE CAUSE	CORRECTIVE ACTION	
	5d. Q2 circuit defective	5d. Check for transitions at Q2 base, emitter, U11-1 and -2, as keyboard is depressed.	
	5e. Q1 circuit defective	5e. check Q1 base emitter	
	5f. loop not connected or insufficient current		
	5g. loop polarity reversed		
	5h. E7 strapped to E8	5h. remove strap	
6. Incorrect response to keyboard	6a. keyboard data lines broken or scrambled		
	6b. keyboard data or strobe inverted		
	6c. keyboard defective		
	6d. improper Baud rate setup		
7. Erratic keyboard response to keyboard	7a. typist operating faster than baud rate		
	7b. Strobe pulse too narrow		
8. Data not placed on screen	8a. loop not connected or insufficient loop current		
	8b. loop polarity reversed		
	8c. U22 circuit defective		*
	8d. Q4 circuit defective		*
	8e. U1-16 circuit defective		*
	8f. U12-U18 defective (one or more devices)		*
	8g. U1 defective		*
9. Incorrect data placed on screen	9a. improper Baud rate setup		
	9b. U12-U18 defective (one or more devices)		*

5-5. FACTORY REPAIR SERVICE.

In the event that difficulty is encountered with this unit, it may be returned directly to MOSTEK for repair. This service will be provided free of charge if the unit is returned within 90 days of purchase. However, units which have been modified or abused in any way either will not be accepted for service or will be repaired at the owner's expense.

When returning the circuit board, place it inside the conductive plastic bag in which it was delivered in order to protect the MOS devices from electrostatic discharge during shipment. (The circuit board must NEVER be placed in contact with styrofoam materials). ENCLOSE a letter containing the following information with the returned circuit board.

Name, address, and phone number of purchaser
Date and place of purchase
Brief description of the difficulty

Mail a copy of this letter SEPARATELY to:

MOSTEK Corporation
Microcomputer Service Manager
1215 West Crosby Road
Carrollton, Texas 75006

Securely package and mail the circuit board, prepaid and insured to:

MOSTEK Corporation
Microcomputer Service Department
1215 West Crosby Road
Carrollton, Texas 75006

5-6. LIMITED WARRANTY.

MOSTEK warrants this product against defective materials and workmanship for a period of **90** days.

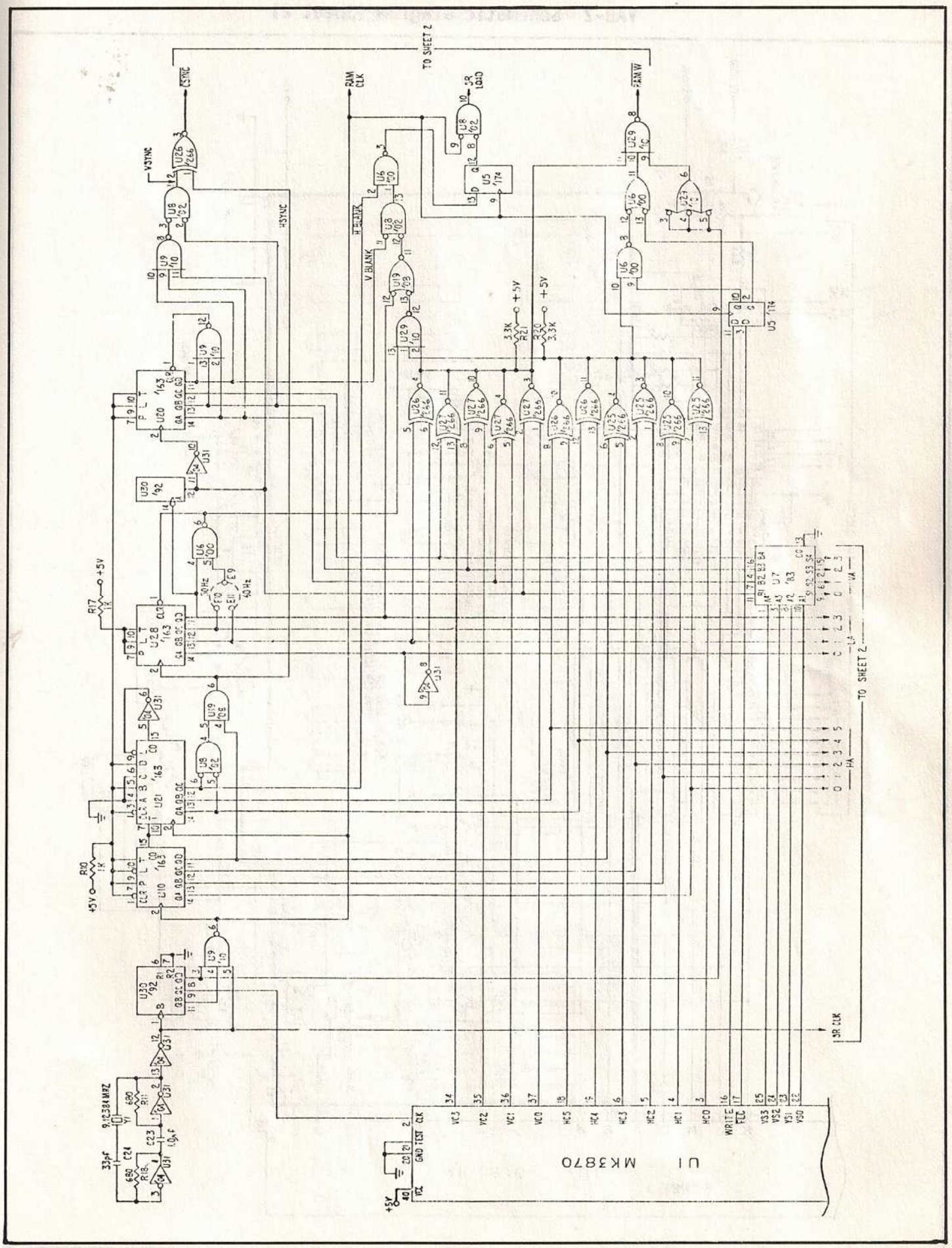
This warranty does not apply to any product that has been subjected to misuse, accident, improper installation, improper application, or improper operation, nor does it apply to any product that has been repaired or altered by other than **MOSTEK** personnel.

There are no warranties which extend beyond those herein specifically given.

APPENDIX A

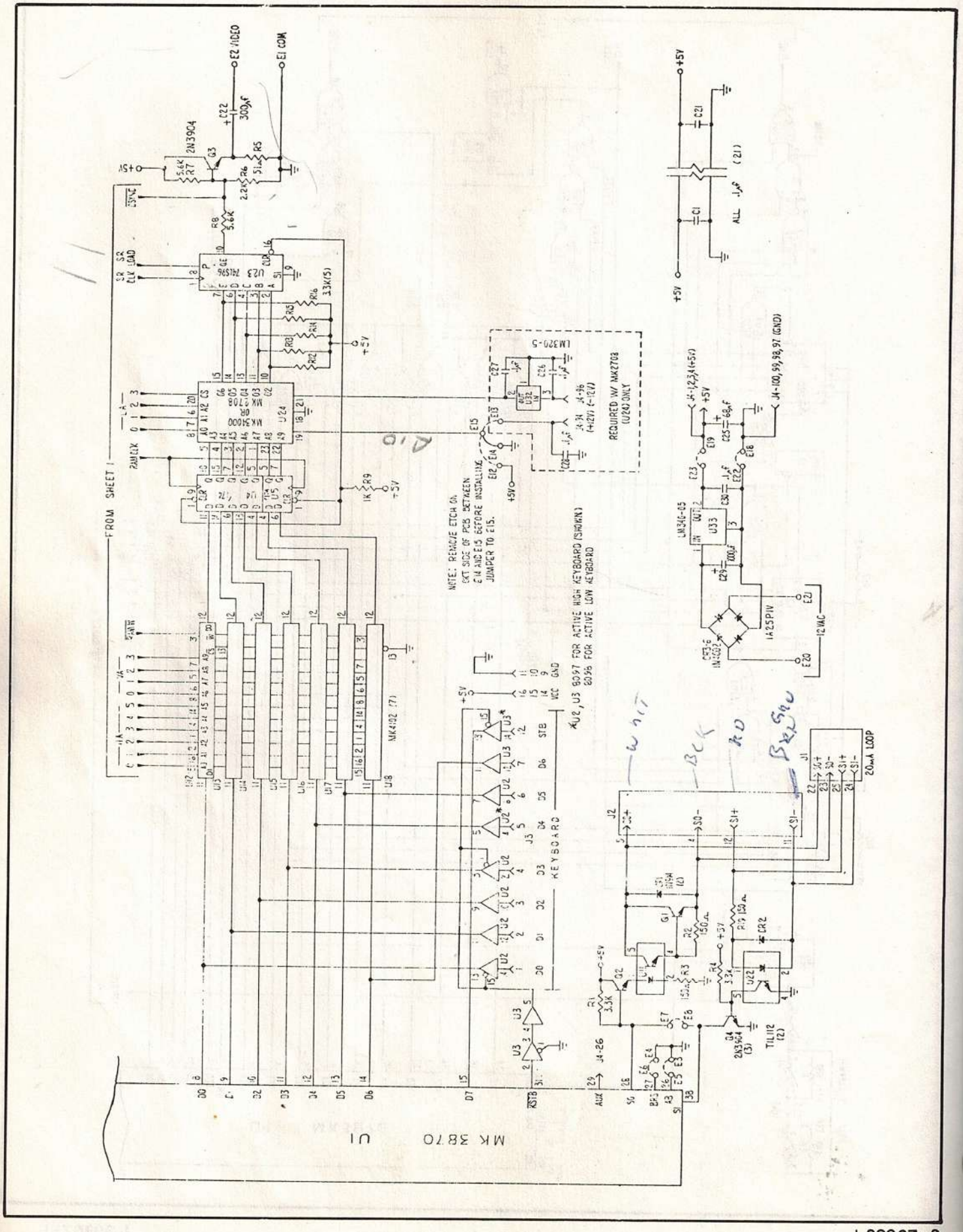
ENGINEERING DRAWINGS AND PARTS LIST

VAB-2 Schematic Diagram (Sheet 1)

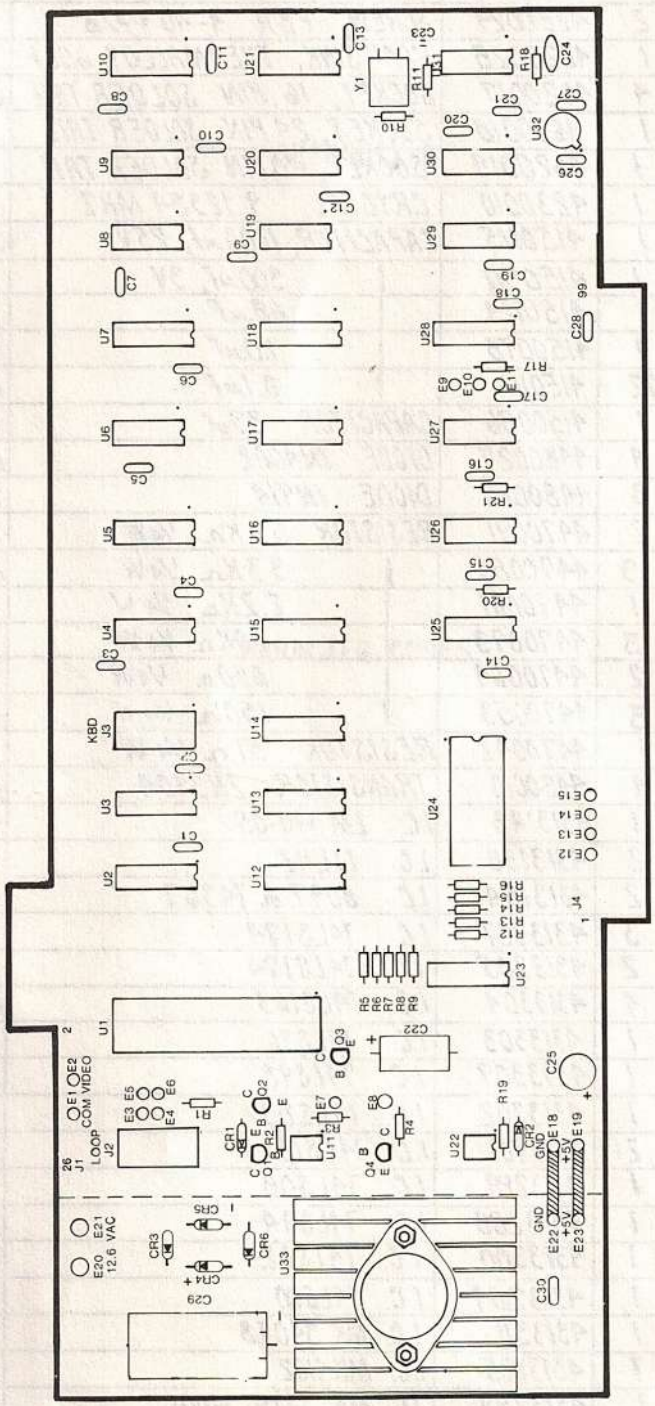


L 99967-1

VAB-2 Schematic Diagram (Sheet 2)



Parts Placement Diagram.



L99966

ITEM	QTY	PART NO	DESCRIPTION	REF. DESIG.	NOTES
47	1	4210040	CONNECTOR, UHF		
46	1	REF	WIRE, 22GA. STANDARD, 3Ft	RED	
45	1	REF	WIRE, 22 GA. STANDARD, 3Ft	BLACK	
44	2	4280021	WASHER, STAR, #4		
43	3	428007	STAKE PIN	E9-11	
42	2	4280030	NUT, HEX, 4-40		
41	2	4280029	SCREW, PBH 4-40 x 3/8		
40	1	4620028	HEAT SINK, THERMALLOY 6061		
39	4	4620017	SOCKET, 16 PIN SOLDER TAIL	J2,3 X2,3	
38	1	4620018	SOCKET, 24 PIN SOLDER TAIL	X24	
37	1	4620019	SOCKET, 40 PIN SOLDER TAIL	X1	
36	1	4230010	CRYSTAL, 9.12384 MHZ	Y1	
35	1	4150115	CAPACITOR, 1000μf, 25V	C29	
34	1	4150110	300μf, 3V	C22	
33	1	4150114	68μf	C25	
32	1	4150078	1.0μf	C23,28	
31	22	4150111	0.1μf	C1-21,30	
30	1	4150086	CAPACITOR 33pf	C24	
29	4	4480027	DIODE IN4002	CR3-6	
28	3	4480026	DIODE IN914	CR1,2,7	
27	2	4470091	RESISTOR 5.6 KΩ 1/4W	R7,8	
26	9	4470085	3.3 KΩ 1/4W	R1,4,12-16,20,21	
25	1	4470081	2.2KΩ 1/4 W	R6	
24	3	4470073	1.0KΩ 1/4 W	R9,10,17,2	
23	2	4470069	680Ω 1/4W	R11,18	
22	3	4470053	150Ω 1/4W	R,2 3,19	
21	1	4470042	RESISTOR 51Ω 1/4 W	R5	
20	4	4480010	TRANSISTOR 2N3904	Q1-4	
19	1	4313143	I.C. LM340-05	U33	
18	2	4313180	I.C. TIL112	U11,22	
17	2	4313154	I.C. 8097 or 74367	U2,3	
16	3	4313307	I.C. 74LS174	U25,27	
15	2	4313305	I.C. 74LS174	U4,5	
14	4	4313304	I.C. 74LS163	U10,20,21,28	
13	1	4313303	I.C. 74LS96	U23	
12	1	4313309	I.C. 74LS92	U30	
11	1	4313302	I.C. 74LS83	U7	
10	2	4313301	I.C. 74LS10	U9,29	
9	1	4313289	I.C. 74LS08	U19	
8	1	4313288	I.C. 74LS04	U31	
7	1	4313300	I.C. 74LS02	U8	
6	1	4313287	I.C. 74LS00	U6	
5	1	4313311	I.C. MK 34073	U24	
4	7	4313185	I.C. MK4102	U12-18	
3	1	4313299	I.C. MK 3870, 14001	U1	
2	REF	450-00190-00	SCHEMATIC VAB-2		
1	1	4610042	P.C. BD., VAB-2		

APPENDIX B

CHARACTER CODES

Table B-1. ASCII/HEX Conversion

HEX CODE	CHAR	HEX CODE	CHAR	HEX CODE	CHAR	HEX CODE	CHAR
00	NUL	20	SP	40	@	60	
01	SOH	21	!	41	A	61	a
02	STX	22	"	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	'	47	G	67	g
08	BS	28	(	48	H	68	h
09	HT	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	S1	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	VS	3F	?	5F	←	7F	DEL

Table B-2. VAB-2 Control Codes And Special Symbols

HEX CODE	SYMBOL	CONTROL	
00	α		
01	β		
02	γ		
03	δ		
04	ε	HOM	Home cursor
05	θ	EOL	Erase to end of line
06	ι	EOS	Erase to end of screen
07	λ		
08	μ	BS	cursor left
09	ν	HT	cursor right
0A	¶	LF	cursor down
0B	Σ	VT	cursor up
0C	ϕ	FF	screen clear
0D	Ψ	CR	cursor to left margin
0E	ω		
0F	Ω		
10	₀	DS	special symbol prefix (down shift)
11	₁	DC1	AUX to logic 1
12	₂		
13	₃	DC3	AUX to logic 0
14	⁰		
15	²		
16	±		
17	÷		
18	≈		
19	√		
1A	∫		
1B	|	ESC	cursor sequence prefix
1C	←		
1D	→		
1E	↑		
1F	↓		
7F	▒	DEL	delete previous character

Table B-3. BAUDOT/HEX Conversion

HEX CODE	LETTERS	CONTROL	FIGURES
00		NULL	
01	E		3
02		LINE FEED	
03	A		-
04		SPACE	
05	S		1
06	I		8
07	U		7
08		CARRIAGE RETURN	
09	D		*
0A	R		4
0B	J		BELL
0C	N		,
0D	F		$
0E	C		:
0F	K		(
10	T		5
11	Z		"
12	L		)
13	W		2
14	H		#
15	Y		6
16	P		0
17	Q		1
18	O		9
19	B		?
1A	G		&
1B		FIGURES	
1C	M		.
1D	X		/
1E	V		;
1F		LETTERS	

Design a low-cost CRT terminal around a single-chip μP

The decreasing cost of processing power makes single-chip μP's ideal building blocks for computer peripherals—especially CRT terminals.

Michael S. Miller, Mostek Corp.

Peripherals now provide the major expense in any new computer system due to the unprecedented rate of cost reductions in CPU's themselves. This price decline could in turn help reduce the cost of peripherals by allowing designers to replace complex and costly controllers with small computers. One such class of peripherals that lends itself to implementation with microprocessors is the CRT terminal.

To be useful, however, a CRT terminal must include the following minimum features:

- 16 lines of 64 characters
- 5×8 dot-matrix character format; upper-and lower-case ASCII
- Page mode from top of screen
- Autoscroll after 16th (bottom) line
- Up, down, right, left and home cursor control
- Carriage return

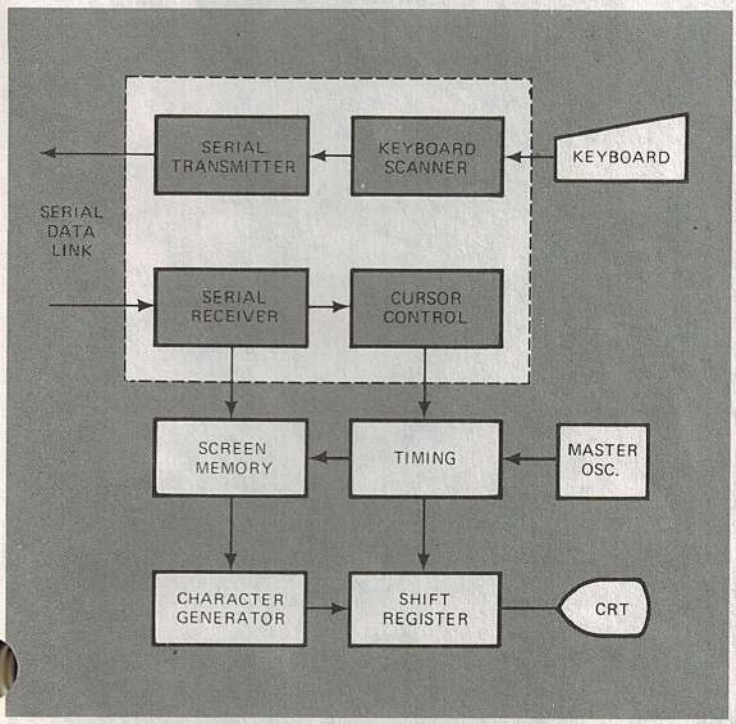
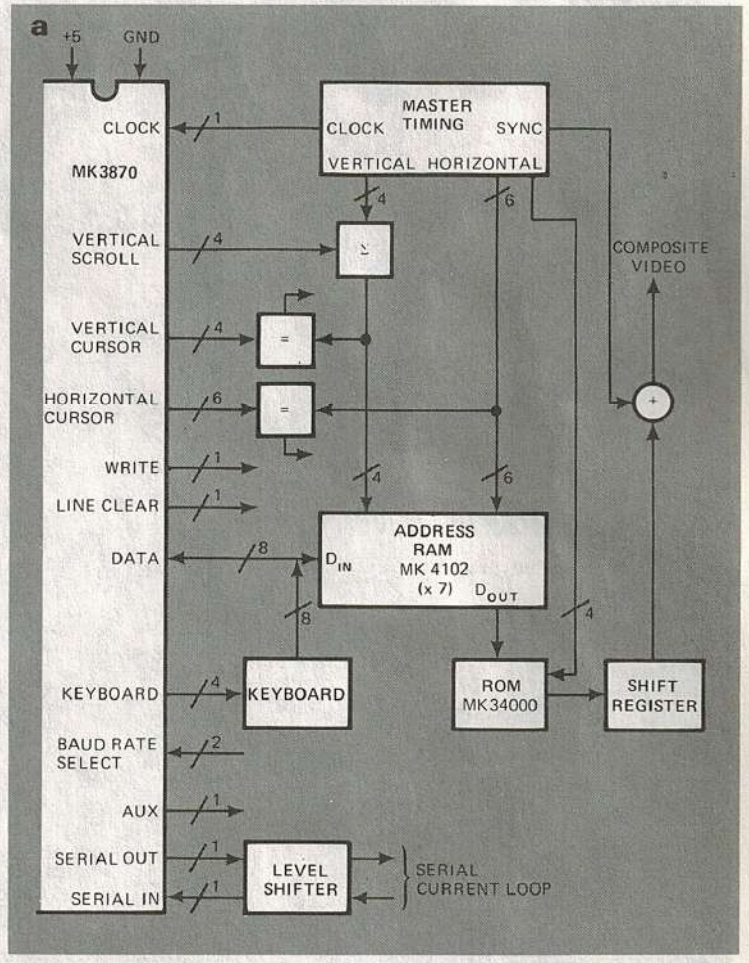


Fig. 1—A single-chip μP can easily replace the SSI and MSI CRT terminal functions outlined in the dotted box.



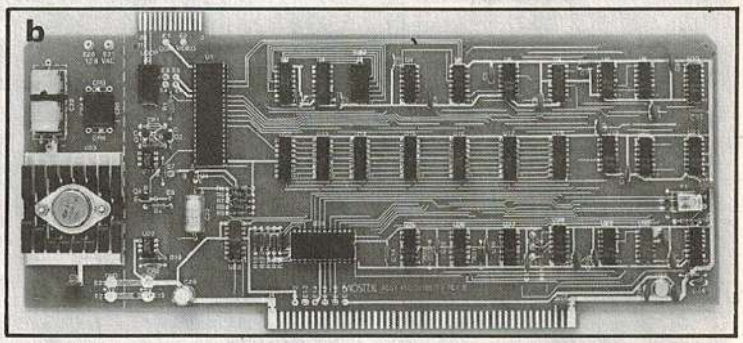


Fig. 2—From a conceptual block diagram (a) to the actual hardware **(b)** is not that long a step when using microprocessors since many hardware deficiencies can be made up in software.

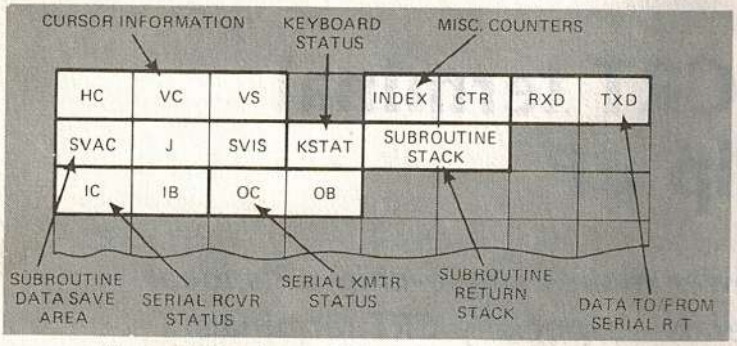


Fig. 3—CPU register-utilization map indicates that only 17 out of 64 registers are used. The extra 47 registers could implement added features such as tab stops, FIFO buffers, etc.

- Screen clear
- Erase to end of screen
- Erase to end of line
- Direct cursor address—absolute or relative
- Data transfer up to 300 baud

What you see, is...

Fig. 1 shows the fundamental building blocks of all terminals. The keyboard scanner monitors the keyboard and, when a key is pressed, passes the corresponding code to the serial transmitter. The

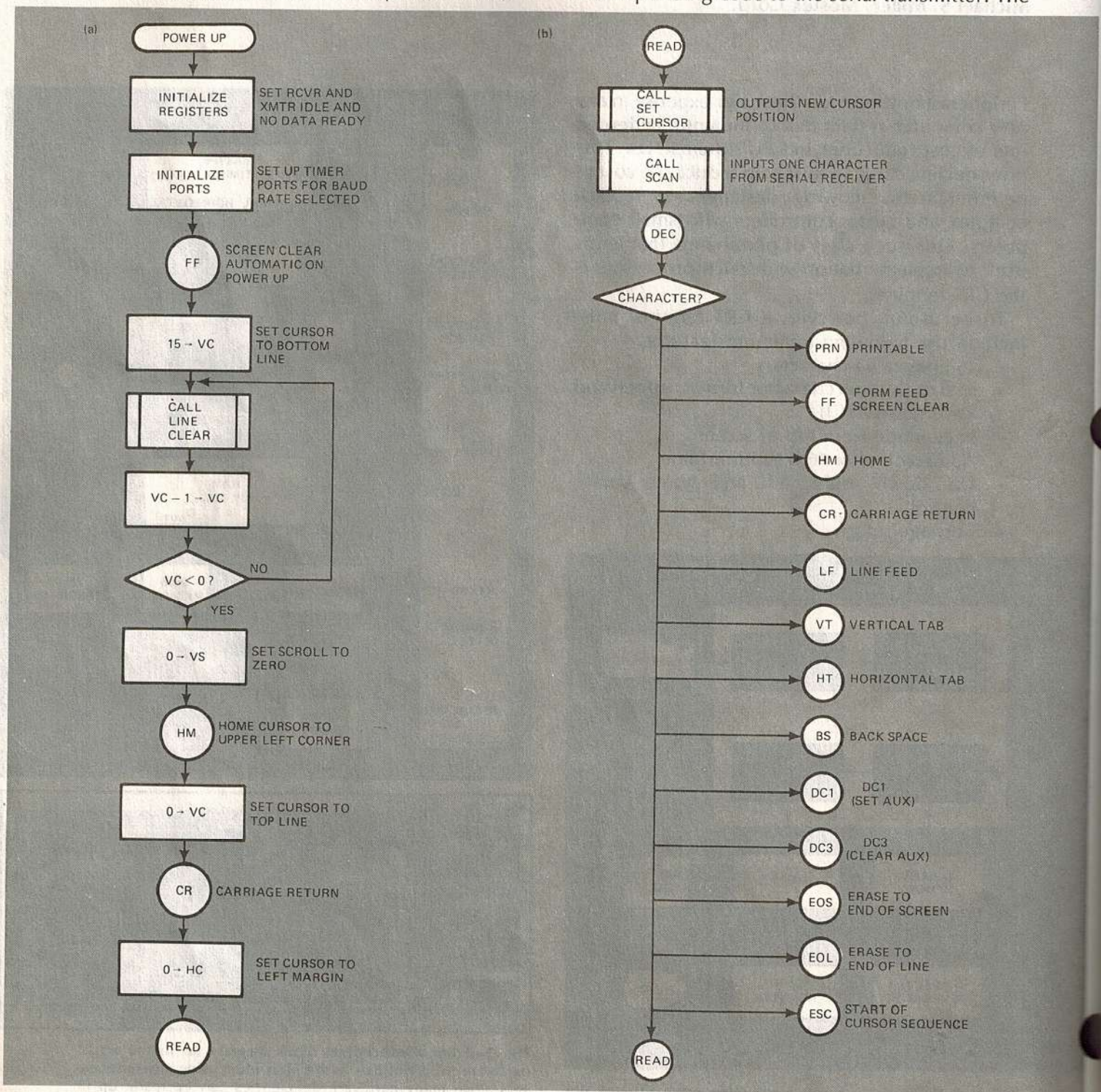


Fig. 4—On power up, the program **(a)** automatically clears the screen and places the cursor in the upper left corner. It then transfers control to the basic READ loop **(b)** which reads one character from the receiver and decodes it.

transmitter converts the parallel data from the keyboard into the appropriate serial format for the data link. Parity generation may be included.

The serial receiver converts incoming serially formatted data into parallel data for processing. Cursor control logic then decodes incoming characters into two groups: printable and special. Printable codes are written into the screen RAM. Special characters are further decoded and their functions executed.

The master oscillator and timing block generate ROM and RAM addresses, composite sync, cursor compare and other control signals. The character generator converts each displayable character into a dot matrix, and the shift register converts the matrices into serial video.

Such a basic CRT terminal **(Fig. 1),** if implemented with standard SSI and MSI, would require >100 packages. An equivalent design using a single-chip μP (in this case the MK 3870) with on-board ROM and RAM to replace the serial data receiver/transmitter, keyboard scanner, control character decoder and cursor control logic requires only one 40-pin, one 24-pin and 27 14- and 16-pin packages: savings of >2/3 the package count.

While this reduced package count would be enough to justify using a μP, further benefits of microprocessorization prove equally impressive. Later versions of the terminal, adding new

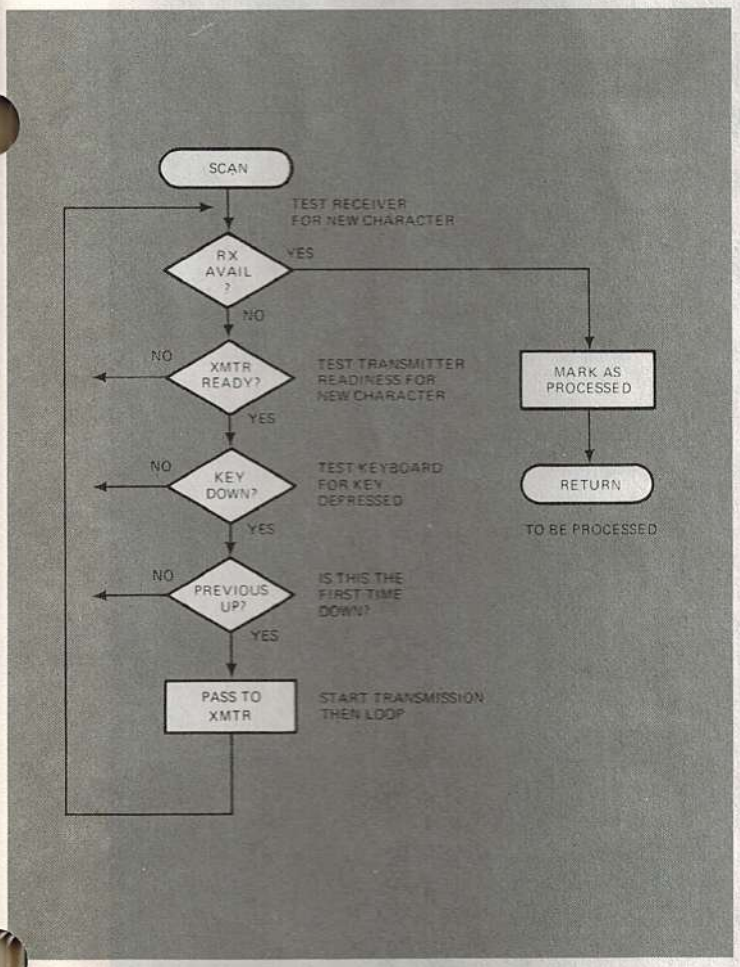


Fig. 5—The SCAN routine alternately tests for receive data ready and transmitter ready. The processor spends the major portion of its time in the loop. It returns to the calling program and places the new character in the accumulator.

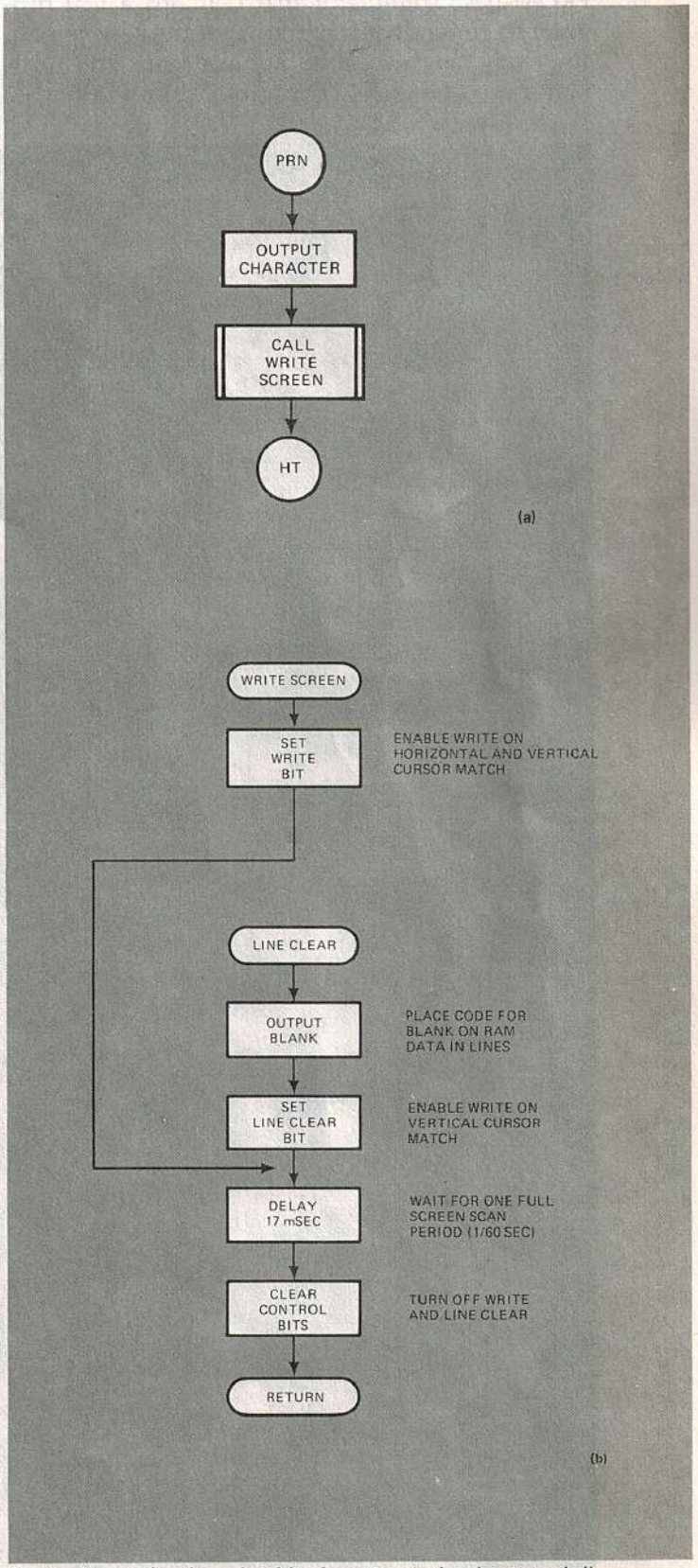


Fig. 6—Processing for printable characters is the shortest of all subroutines. Note how the WRITE SCREEN and LINE CLEAR subroutines share part of the same program.

features, require only a new μP-ROM pattern, keeping pc-board layout intact. Field modifications are literally a snap—snap out the old μP, snap in the new. Field service typically entails temporary replacement of the processor with a PROM-based emulator that contains a test program to generate test patterns for the serviceman. In really high-volume applications, the test-pattern generator could be simply another single-chip μP.

"Blocking" out a solution

The detailed block diagram **(Fig. 2a)** shows the MK 3870 pin allocations and their relations to the rest of the circuitry. Timing for the terminal derives from a TTL divider chain that generates RAM and ROM addresses, composite sync, processor clock and other control signals. Additionally, the vertical address from the timing chain adds to the vertical scroll address to generate the physical RAM vertical address. This allows the

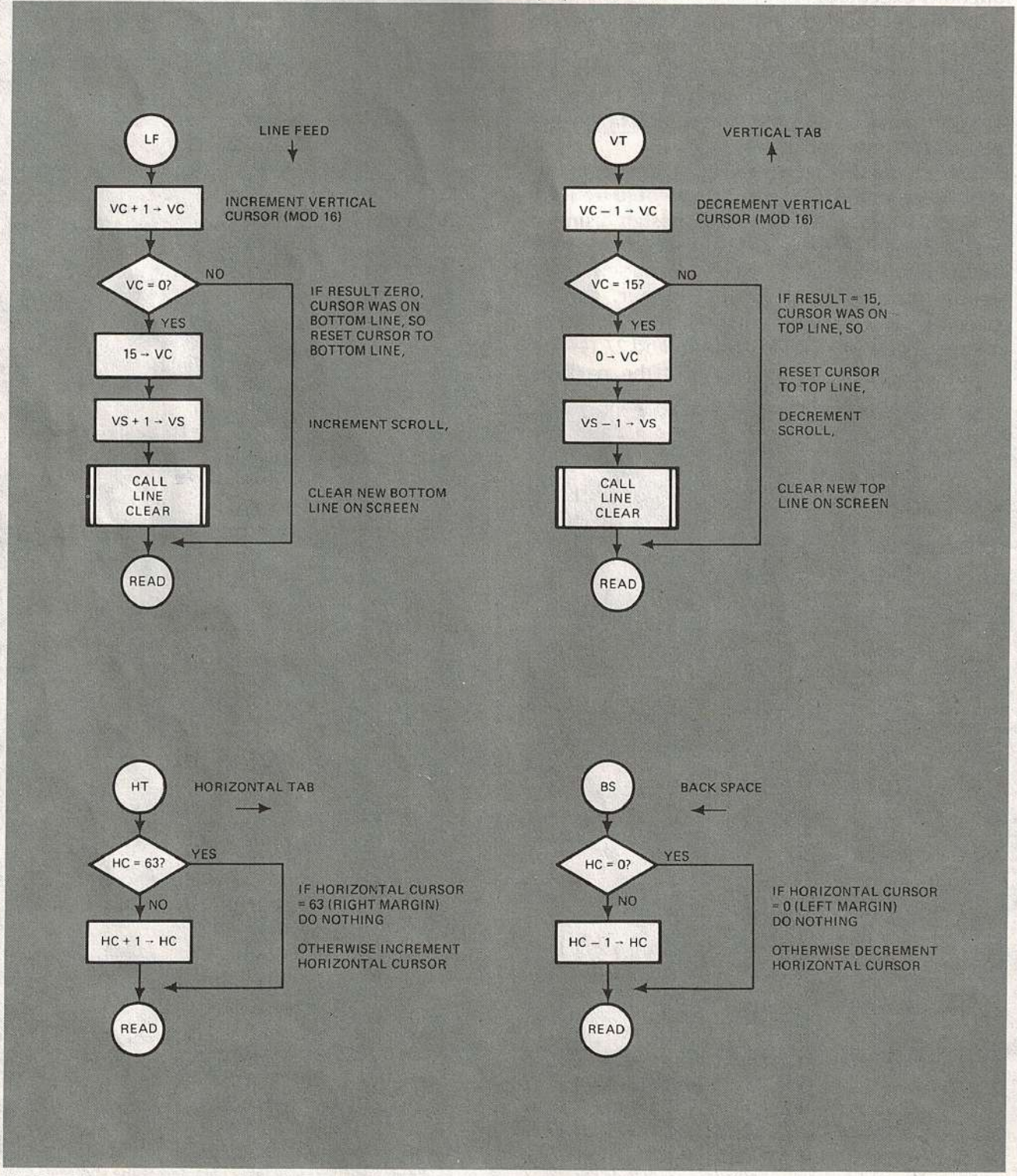


Fig. 7—Cursor moves use modulo arithmetic for autoscrolling in the LF and VT routines, while in the HT and BS routines nothing happens when a margin limit is reached.

displaying of any physical line of the RAM as the top line on the screen.

Seven 1k static RAM's constitute the screen refresh memory for 1024 characters (16×64). Screen RAM updating by the processor can occur only when the cursor address equals the RAM address. While this approach limits the data-transfer rate (from processor to screen) to the ac-line frequency (50-60 Hz), it also reduces cost and complexity by eliminating address multiplexers.

Adding some "character" to your ROM's

This setup uses an MK 34000 Series 16k ROM as the character generator. The circuit actually requires just 8k. This leaves half the ROM blank, or it allows two complete and different strap-

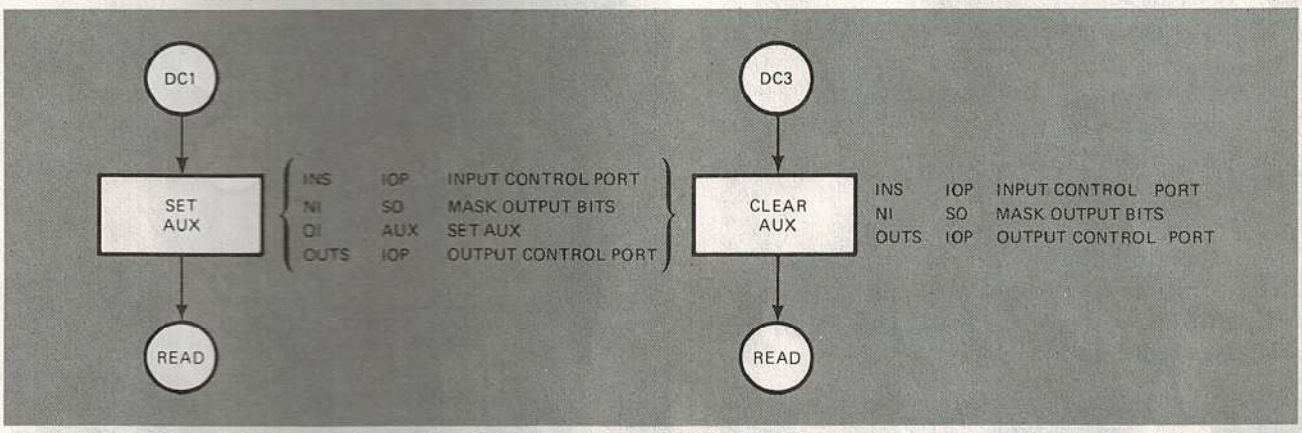


Fig. 8—Since DC_1 and DC_3 set and clear the AUX pin on the MK 3870, this pin can be used to control a tape drive, change character sets and indicate status.

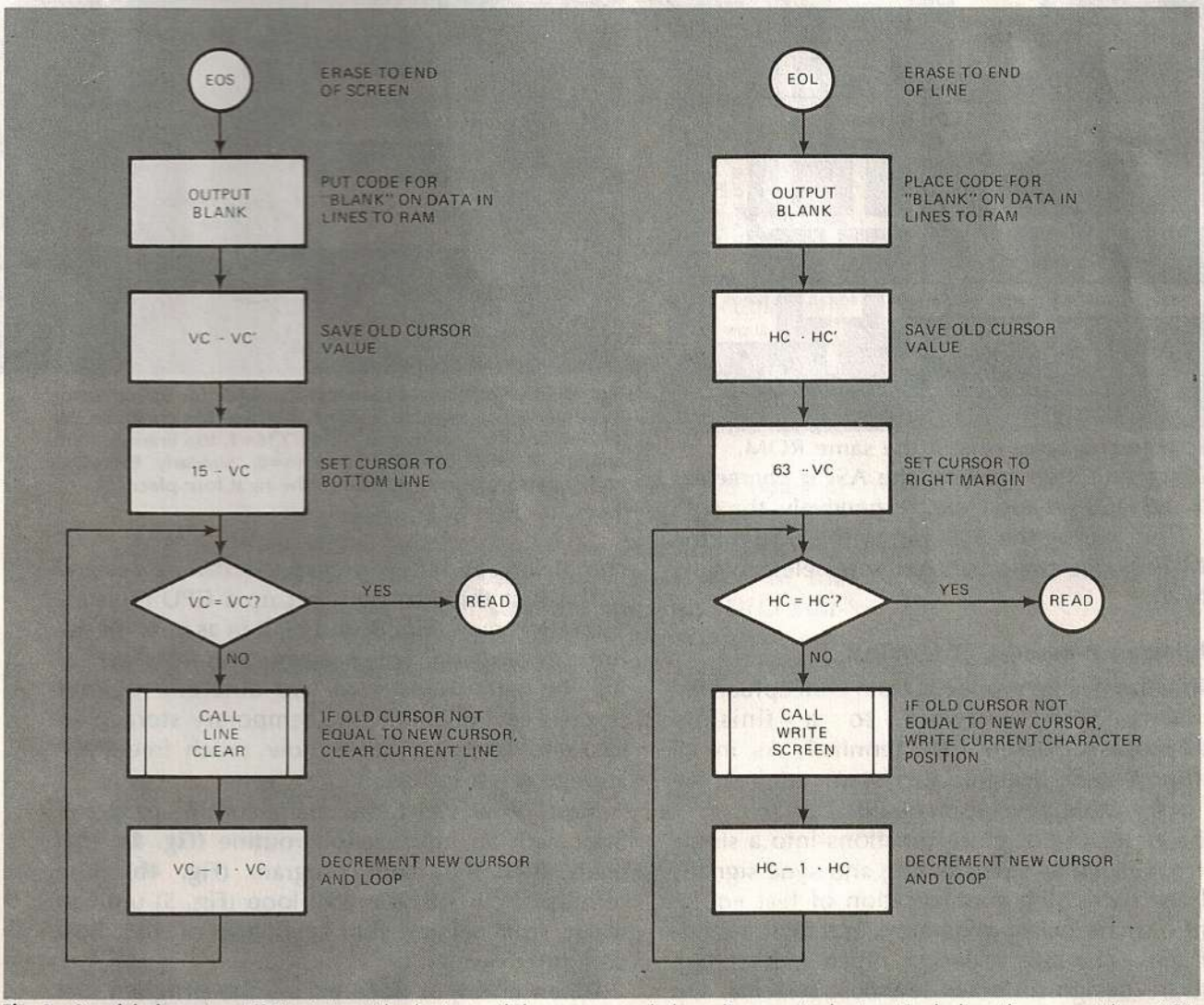


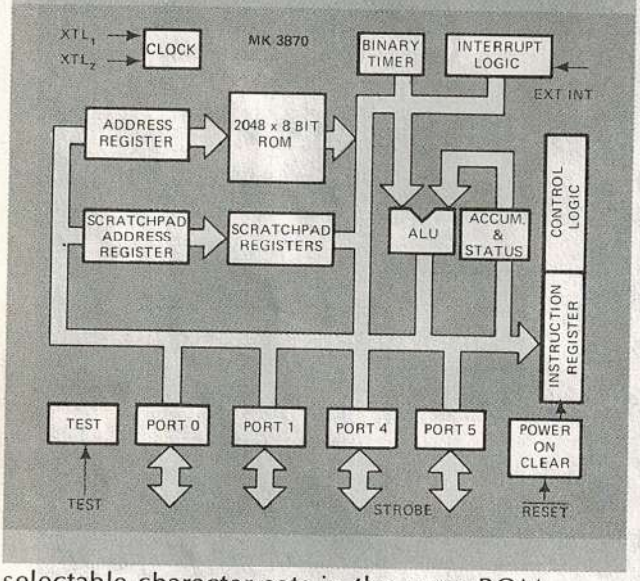
Fig. 9—Special character, EOS, starts at the bottom of the screen and clears lines up to but not including the current line. EOL starts at the right margin and clears up to but not including the current character position.

Features of the MK 3870

The accompanying block diagram shows the general architecture of the MK 3870; resources of the MK 3870 single-chip μC include:

- 2k bytes ROM
- 64 bytes RAM
- 32 bidirectional I/O lines (four 8-bit ports)
- External interrupt input
- Over 70 instructions
- 8-bit binary counter with seven selectable prescalar values
- 2 μsec basic register-to-register instruction time.

The external interrupt can also be used as a general-purpose input, event counter or for pulse-width measurement.



selectable character sets in the same ROM.

Going one step further, the ASCII characters DC1 and DC3 set and clear, respectively, the AUX pin. Thus, wiring the AUX pin to the MSB on the ROM provides two character sets selectable by software.

Building a μP-based CRT terminal

Now the design proceeds from conceptualized hardware and software to a finished microprocessor-based CRT terminal. As in all microprocessor designs, this step requires the use of available development aids.

The trend to put more functions into a single LSI chip (such as video timing and sync signals) and the increasing sophistication of test equipment ("smart" word generators and logic analyzers) make hardware design much easier than software design for small low-cost systems. On the other hand, such systems tend to replace hardware with software, which increases software

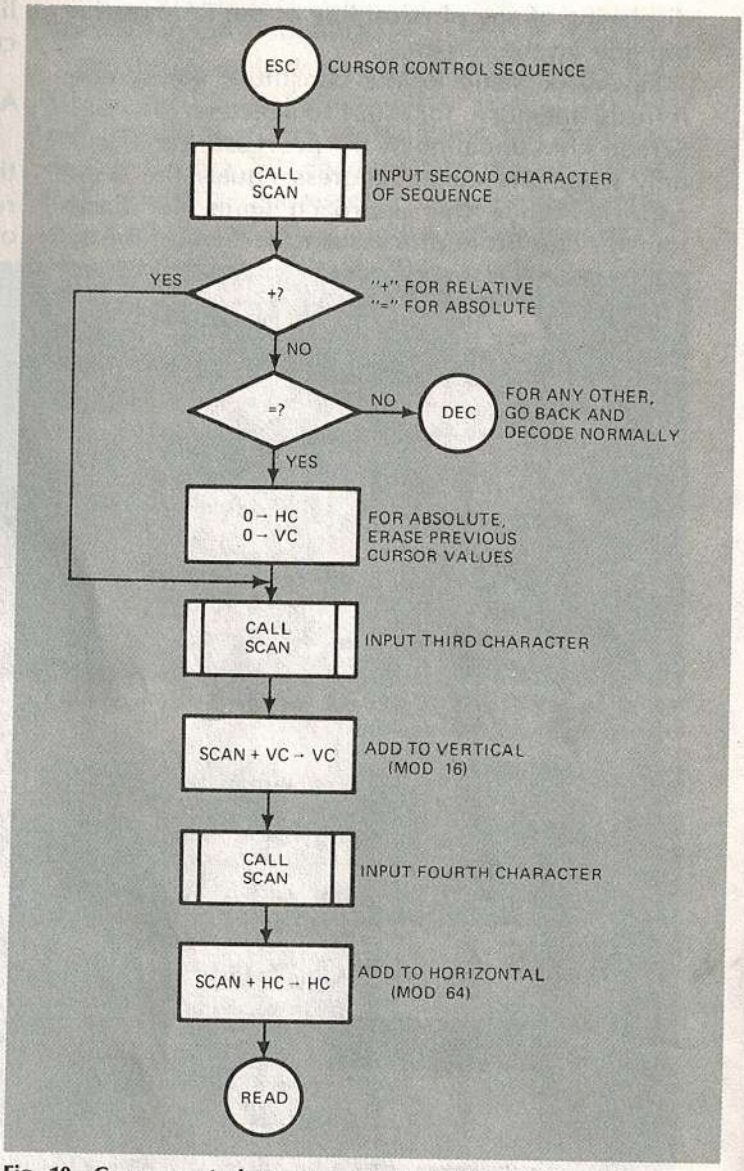


Fig. 10—Cursor control sequences provide for direct cursor movement. For example, ESC=A,B moves the cursor to the second line (ASCII value of A MOD 16=1, top line=0), third column (B MOD 64=2, left column=0). Similarly, ESC+C,D moves cursor down three and to the right four places.

complexity. So let's turn now to software design.

The first thing to do is set up a CPU register-utilization map **(Fig. 3)** and fill it in as you "flesh" out the program. This prevents you from assigning the same register to two different routines (except as nonconflicting temporary storage areas) and lets you know how much free RAM storage is left to use.

Next, flow chart the main control program. Start with an initialization routine **(Fig. 4a)** that leads into the main program **(Fig. 4b).** The terminal now sits in a wait loop **(Fig. 5)** until an input from either the keyboard or the host computer occurs.

When an input does occur, the program first checks to see if it's a printable character and branches to the appropriate subroutine **(Fig. 6).** If

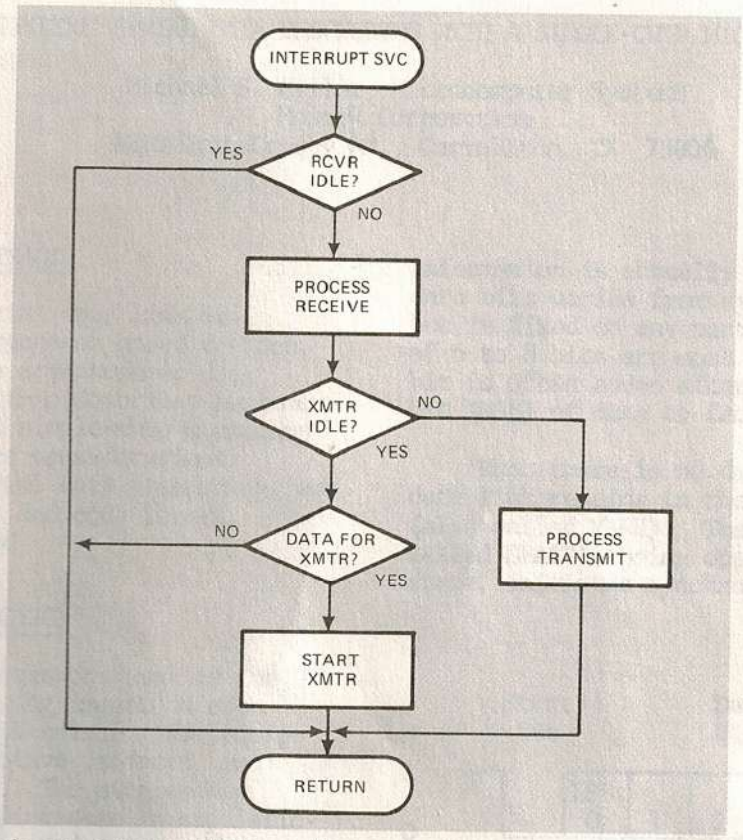


Fig. 11—Software UART provides communications between the outside world and CRT terminal.

nonprintable, the program then goes down its list and compares the received character to the cursor control commands **(Fig. 7)**, special function commands **(Fig. 8)** and erase commands **(Fig. 9)**.

To fulfill the last two requirements for the minimum terminal configuration, you need a cursor sequencer routine **(Fig. 10)** and a software UART **(Fig. 11)**.

The final program required only 475 bytes of program storage (23% of the 2k available) and 17 bytes of RAM (26% of the 64 bytes available). Many other features could be added in the remaining space: software tab stops, FIFO buffers, nonstandard code conversions and processor-encoded keyboard, to name a few. □

Reprinted from EDN May 5, 1977

ASYNCHRONOUS SERIAL DATA PROCESSING WITH A SINGLE-CHIP MICROCOMPUTER

Michael S. Miller, Microcomputer Systems
Mostek Corporation
1215 West Crosby Rd., Carrollton, TX 75006

ABSTRACT

An asynchronous serial data link is required in many microprocessor based devices. This paper first reviews asynchronous data format definitions, and then describes implementation using the MK 3870 single-chip microcomputer. Hardware interface considerations, software schemes for serial data processing, use of timer and interrupts, and code format conversions are discussed.

INTRODUCTION

Single-chip microprocessors such as the MK 3870 are having a growing impact on such devices as computer peripherals,[1] test equipment, consumer and automotive products, and communications equipment. The single-chip microprocessor allows an increase in sophistication with concurrent cost reduction. Many applications are finding need of communications outside the particular device. Such communication is being made practical by the benefits inherent in microprocessor designs. This paper discusses one form of communication - asynchronous serial data links - and how a single-chip microprocessor may be used to implement such a design.

SOME PRELIMINARY DEFINITIONS

Before jumping into the details of a microprocessor design, it would be beneficial to review some basic definitions related to asynchronous data links.

Asynchronous Serial Data Format

The asynchronous serial data format was developed to facilitate data transmission between mechanical devices.[2] Since its inception, the format has been expanded into use for many low to medium speed data communication tasks. The asynchronous format is characterized by the inclusion of information intended to facilitate the synchronization of the two ends of the data path.

As shown in Figure 1, a "frame" of data consists of a start bit, data bits, and a stop bit. The start bit and each data bit are of equal length. The stop bit is typically between one and two units in length. It is common to refer to a stop bit whose length is two units as being 2 stop bits, although only one bit of information is actually conveyed. The number of data bits varies from application to application, but is fixed on any particular link. Data words of 5 to 8 bits are most common. An extra parity bit is often added after the most significant bit (MSB) of data to facilitate error detection.

When there is no data being transmitted, the data link remains in the stop bit or idle state (also called MARK). The start bit state (also called SPACE), being opposite to the stop bit state, initiates synchronization.

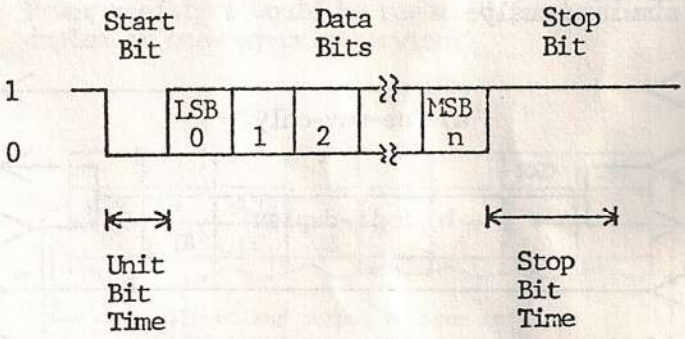


Figure 1

Data Transmission Rates

Several terms are used to refer to the rate of transmission across a data link:

1. Words Per Minute (WPM): The number of words transmitted per minute (one word equals 5 or 6 characters). Not precise.

2. Characters Per Second (CPS): The number of characters (frames) transmitted per second. May be computed by reciprocating the length of time from the beginning of the start bit to the end of the stop bit.

3. Bits Per Second (BPS): The number of bits transmitted per second. Computed by multiplying the number of bits per character by the number of characters per second.

4. Baud: Baud is defined to be the "unit of modulation rate."[3] Baud is computed by reciprocating the length of the shortest bit (the start or data bits).

Baud is becoming the most prevalent unit of measurement for data transmission rate. Table 1 lists the most used baud rates.

Mechanical Devices	Data Links	
45.45	300	2400
74.2	600	4800
110	1200	9600

Table 1: Common Baud Rates

Data Links

Data links may be classified according to the directionality of data transmission. One way only links are capable of data transmission in one direction only; that direction is fixed. Half-duplex links are capable of transmission in either direction, but not simultaneously. Full-duplex links transmit in both directions simultaneously.

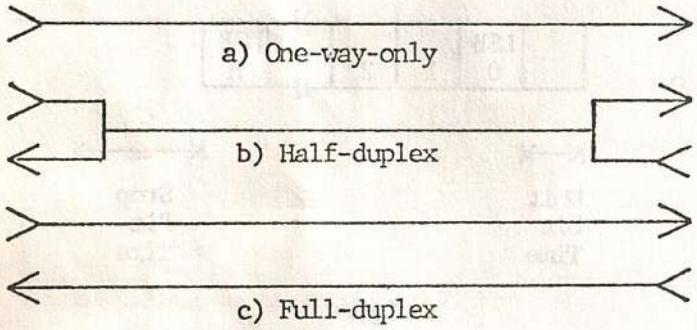


Figure 2 - Data Link Types

Information Codes

Many codes have been devised for representing information. The Morse telegraph code is an example of an early code used for manual transmission of data. More recent codes include the 5 bit Baudot code, the 6,7, and 8 bit ASCII code, and the EBCDIC code. Baudot is commonly used in older mechanical printers in military, telegram, and news service. ASCII has become an almost universal code for computer data. ASCII stands for American Standard Code for Information Interchange. EBCDIC is used primarily by IBM.

HARDWARE INTERFACE

The exact hardware interface between the microprocessor and the data line will be highly application-dependent. Figure 3 presents several possibilities. Differential driver/receivers (a) and twisted pair lines are recommended for long lines; simple TTL gates (b) may be sufficient for a few feet at low data rates. Driver/receivers for direct EIA RS-232-C interface are available. Current loop interfaces are often used with mechanical devices. Typically, the computer end of the circuit is a non-isolated current loop (c), and the terminal end is an isolated loop (d). Several isolated current loop circuits may be wired in series for party line circuits or half-duplex service.

Clock Rate Considerations

Factors involved in clock rate considerations are:

(1) CPU clock rate	f
(2) data link baud rate(s)	B
(3) slice factor	s
(4) timer divisor factor	d

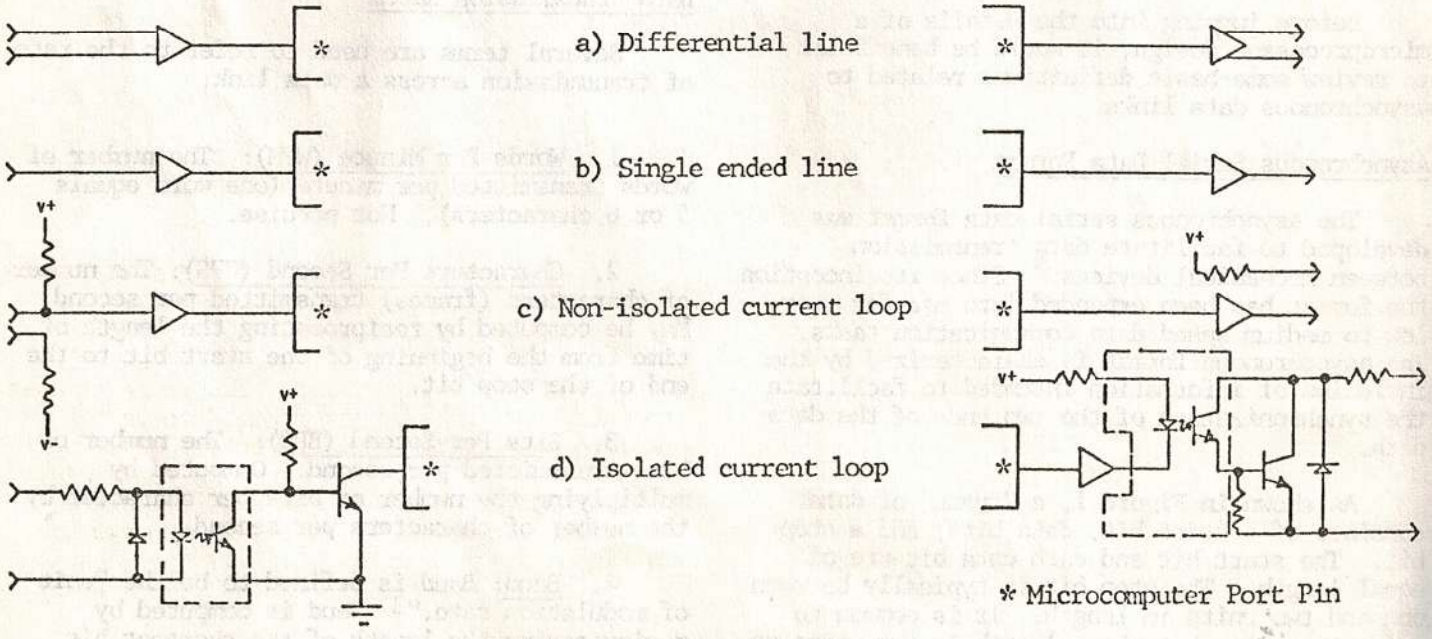


Figure 3 - Line Interfaces

1. CPU Clock Rate is often determined by factors not relating to the software UART. For example, a 3.58 MHz color TV crystal may be selected for availability and low cost. Or an existing clock source may be used. Generally, flexibility in the timer will allow great latitude in CPU clock rate selection.

2. Baud Rate is often determined by the application. Factors such as hardware limitations may influence maximum practical baud rate.

3. Slice Factor refers to the number of samples per bit time. Hardware UARTs typically use a slice factor of 16 (also referred to as 16 X clock). A large slice factor improves receiver reliability at the cost of more CPU time spent in the UART routine. A slice factor of 1 is entirely adequate for transmitting, and for one-way-only or half-duplex reception. Full duplex reception will generally require a slice factor of 8 or greater.

4. Timer Divisor Factor is calculated from the preceding factors according to the formula

$$d = \frac{f}{sB} \qquad (1)$$

where f is the effective CPU cycle rate (or input to timer prescaler). The value computed for d will seldom be an integer value directly attainable from a timer (unless the CPU clock rate is selected for the UART).

For example:

A given application requires a CPU cycle rate of 1.52064 MHz. A slice factor of 8 is selected.

Table 2 shows computed values of d for four baud rates, along with nearest divisor attainable with the MK 3870, and the resultant baud rate error. Since a slice factor of 8 provides for several percent of error margin, the results of Table 2 are entirely adequate.

f	s	B	d	nearest integer	error
1.52064 M	8	300	633.6	635	-0.22%
		110	1728.0	1730	-0.12%
		74.2	2561.7	2560	+0.07%
		45.45	4181.8	4180	+0.04%

Table 2 - Divisor Computations

A SOFTWARE UART

The task of servicing a serial data line is an ideal one for a microprocessor such as the MK 3870. Flexible I/O, timer, and interrupt structure greatly simplify the software UART task while keeping hardware costs to a minimum. Once the software UART has been started, the overhead software to pass data to and from it is actually less than if a hardware UART had been used.

Resource Allocations

As in any computer project, the software UART requires careful resource allocation planning.

1. CPU Registers. Of the 64 registers available in the MK 3870, nine are required for a basic full duplex UART. Other registers might be required for code conversions, etc. Fewer registers would be required for half-duplex or one-way-only service.

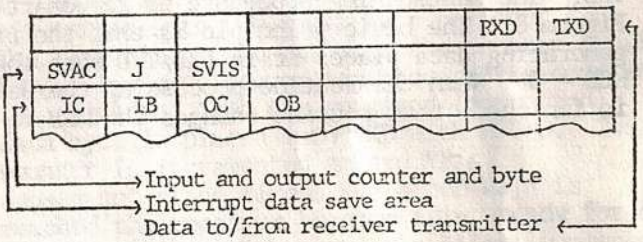


Figure 4 - CPU Register Map

2. I/O Ports. In the example to be discussed, four bits of I/O are used:

2 bits baud rate select (1 of 4 code)
1 bit serial input
1 bit serial output

3. Timer & Interrupts. The timer is used to interrupt the main program and cause the UART to sample the input and output bits.

4. Memory Map. The MK 3870 has fixed interrupt vectors. The resultant memory map is Figure 5.

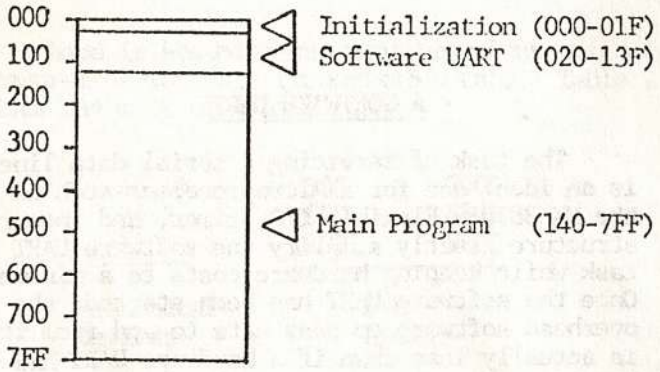


Figure 5 - ROM Map

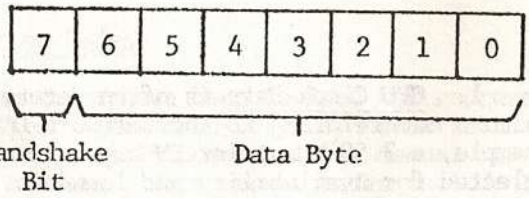


Figure 7 - RXD/TXD Format

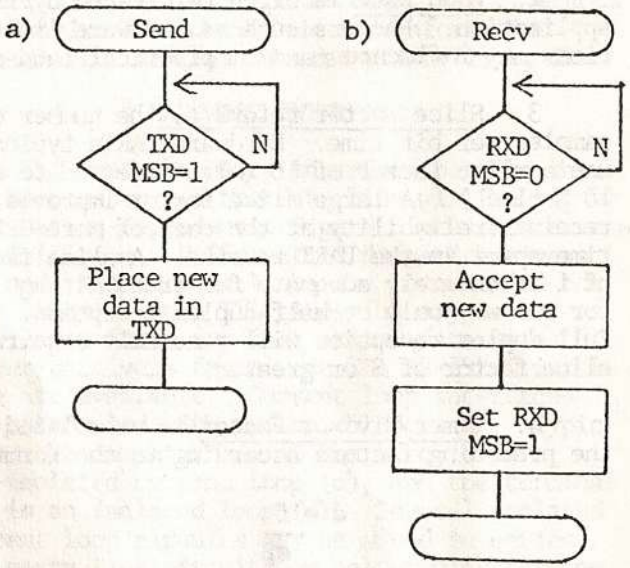


Figure 8 - Handshaking Routines in Main

Overall Structure

The overall structure of the system is flowcharted in Figure 6. Figure 6a shows that, after the UART is initialized and interrupts are enabled, the MAIN program UART (6b) is completely autonomous, communicating TXD. MAIN may consider RXD and TXD to be hardware ports with automatic handshaking. Figure 7 shows the format of RXD and TXD. The handshaking procedure is flowcharted in Figure 8. The basic principle is that the routine generating data places it in RXD/TXD with the MSB = Ø. When the routine processing the data is finished, that routine changes the MSB to 1.

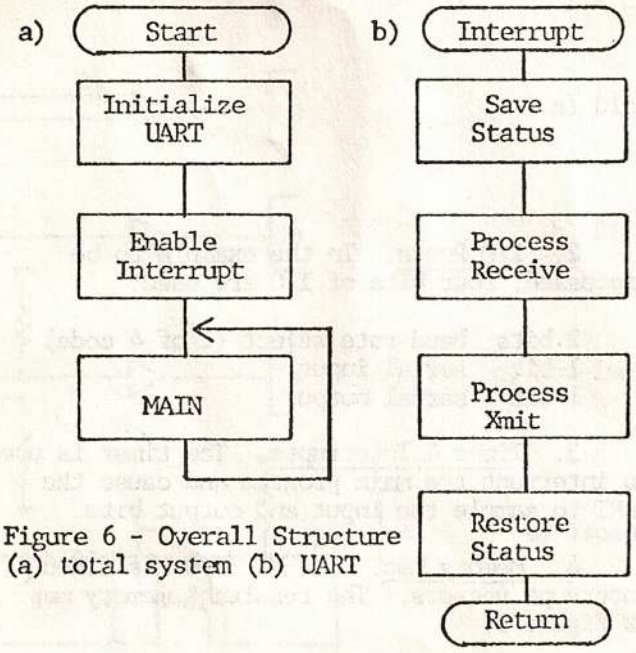


Figure 6 - Overall Structure
(a) total system (b) UART

Starting the UART

Initialization of the UART involves setting up the port bits, timer, and UART registers.

Port bits: The serial out pin is initialized to a "MARK" state. The other pins are cleared for use as inputs.

Timer presets are determined by the baud select pins. In the example, one of the four baud rates listed in Table 2 is selected by the 2 bit code. From Table 3 the prescaler and divisor values are selected and output to the timer control (6) and timer count ports (7), respectively.

d	prescale	divisor	Port 6	Port 7
635	5	127	4A	7F
1730	10	173	6A	AD
2560	40	64	AA	40
4180	20	209	8A	D1

Table 3 - Timer Setup

All UART registers are set to FF_{16}. RXD and TXD are thus initialized to a "no data available" state, and the slice counters are set to the idle state.

Interrupt Service

In the MK 3870, when an interrupt is acknowledged, the program counter (PØ) is saved into the stack register (P) and the vector (Ø2Ø for timer, ØAØ for external) is placed into PØ. Other register saves are accomplished at the interrupt routine, to allow only those registers disturbed to be saved. Figure 9a shows the sequence at the interrupt routine entry point. The accumulator (A), status word (W), and indirect scratchpad pointer (IS) are saved into three CPU registers (SVAC, J, SVIS respectively) dedicated to this purpose (see also figure 4). Figure 9b shows the restoration of those registers and re-enabling of interrupts before returning to MAIN.

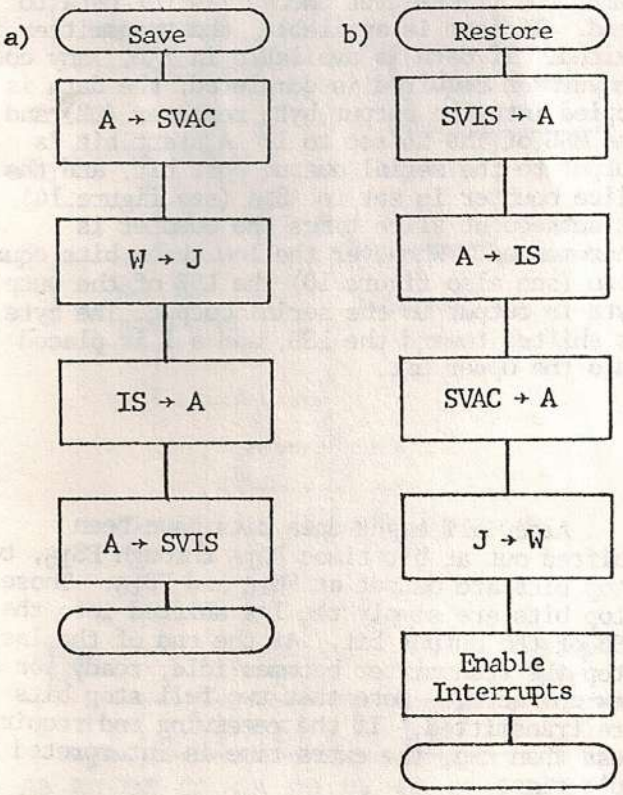


Figure 9 - (a) Save Status (b) Restore Status

The Receiver

Figure 11 is an overall flowchart of the receiver. When no data is being received, the receiver is idle. The receive counter (IC "input counter" of figure 4) contains FF_{16}. Figure 10 shows the format of the slice counters. At each slice time, the receiver detects the idle FF_{16} and tests for a start bit. If a start bit is not detected, the slice count is reset to FF_{16} and the receiver is exited. When a start bit is received, the slice count is set to $B4_{16}$ (see figure 12). At each subsequent slice time, the count is incremented. Four slices later, when the count becomes $B8_{16}$, the receiver again checks for a start bit. If the start bit is no longer there, the receiver assumes a false start bit and resets the counter to idle. If the start bit is still good, the receive process is allowed to continue.

At subsequent slice times the counter is incremented. Each time the low 3 bits are all Ø, the serial input line is sampled and the new bit merged into the input byte register (IB). The last bit is sampled when the counter equals $F8_{16}$. Any code conversion required is done at this time, and then the received (converted) data byte is placed into RXD with MSB=Ø. The counter is incremented toward FF_{16} at subsequent slice times, and after FF_{16} is reached the receiver becomes idle, ready for a new start bit. Note that only slightly more than one half stop bit is required for proper reception. This helps guarantee that the receiver can keep up with the transmitter even if the transmitter is operated a slightly faster than nominal baud rate, or in the presence of data link distortion.

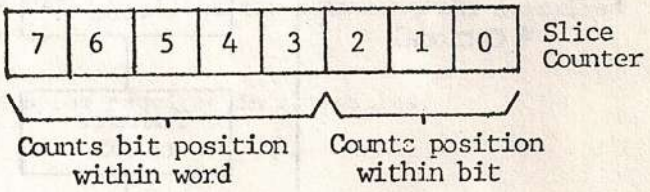


Figure 10 - Slice Counter Format

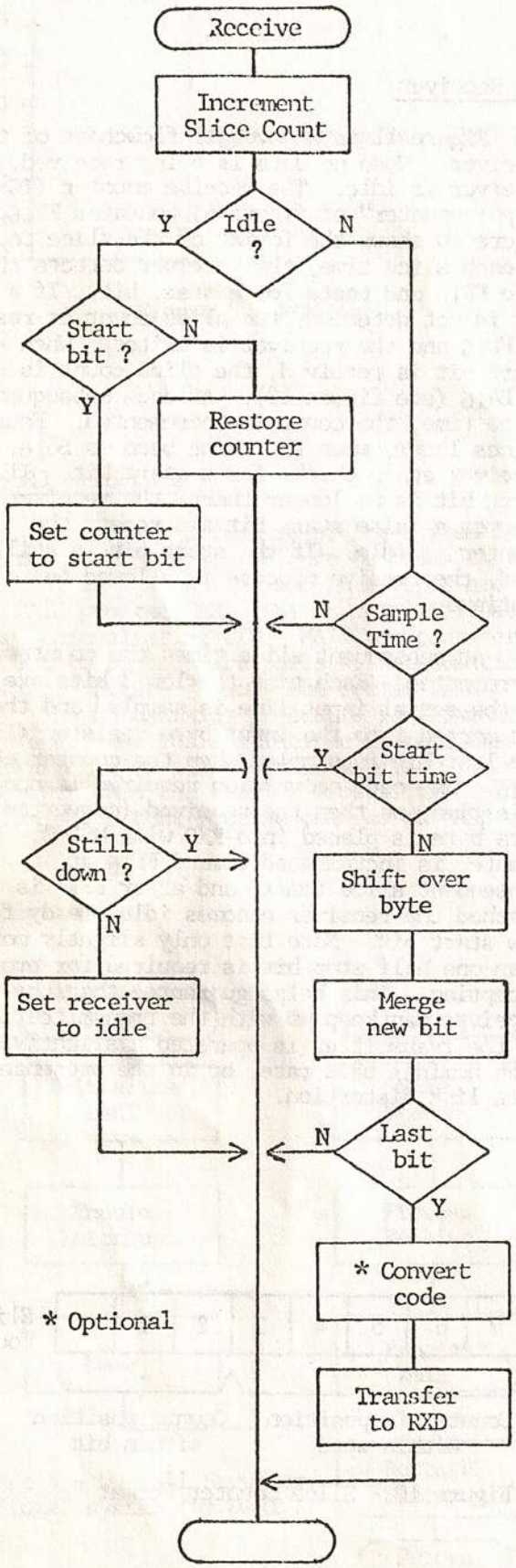


Figure 11 - Receive Processing

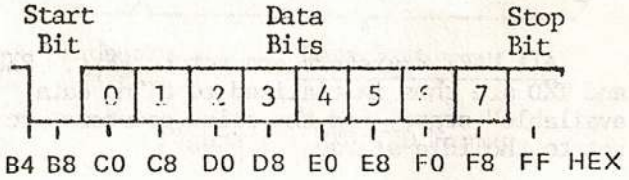


Figure 12 - Receive

The Transmitter

Figure 13 is the flow chart of the transmitter. When no data is being transmitted, the transmitter is idle, and the slice counter (0C of figure 4) contains FF_{16}. At each slice time, the transmitter checks TXD for data to send. If none is available, the transmitter is exited. If data is available in TXD, any code conversion required is completed, the data is copied into the output byte register (0B) and the MSB of TXD is set to 1. A start bit is output to the serial output port bit, and the slice counter is set to $A8_{16}$ (see figure 14). At subsequent slice times the counter is incremented. Whenever the low three bits equal zero (see also figure 10) the LSB of the output byte is output to the serial output. The byte is shifted toward the LSB, and a 1 is placed into the upper bit.

After all eight data bits have been shifted out at bit times $B0_{16}$ through $E8_{16}$, two stop bits are output at $F0_{16}$ and $F8_{16}$. Those stop bits are simply the 1's shifted into the MSB of the output bit. At the end of the last stop the transmitter becomes idle, ready for a new character. Note that two full stop bits are transmitted. If the receiving end requires less than two, the extra time is interpreted as idle time.

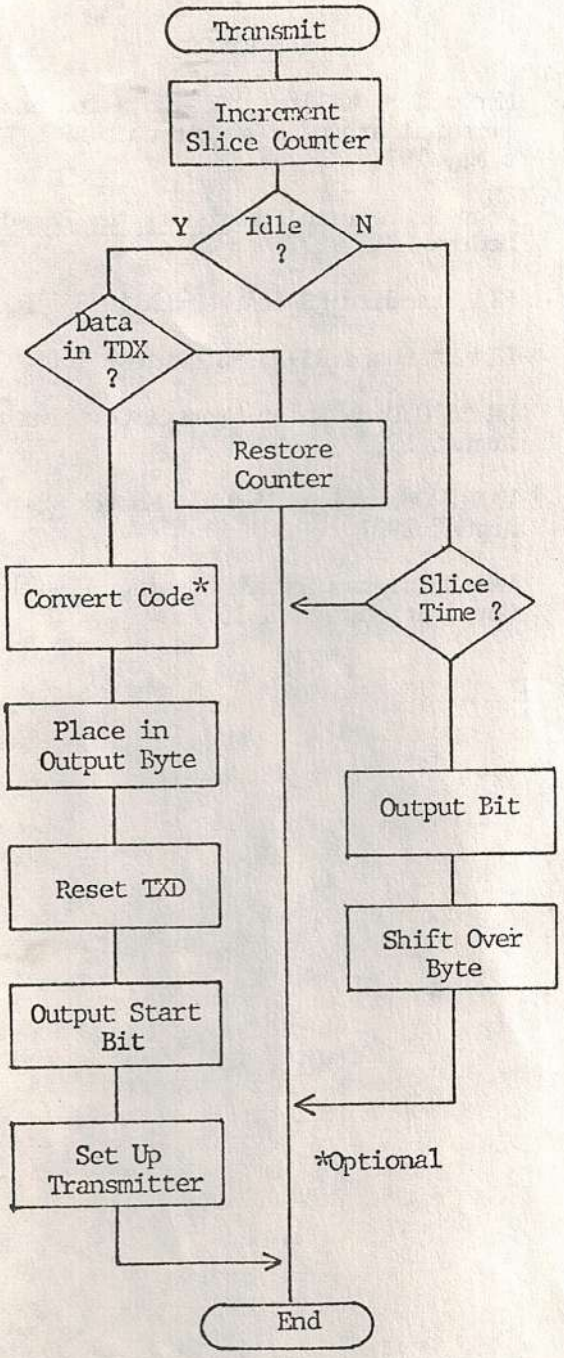


Figure 13 - Transmit Processing

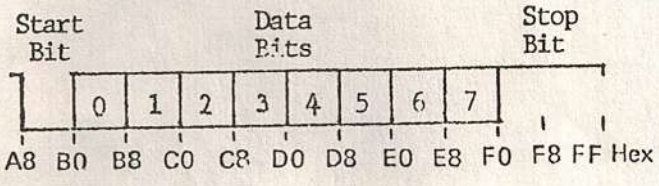


Figure 14 - Transmit

One-Way-Only Processing

Software for a one-way-only processor is much simpler than the full duplex UART described above. (Half duplex may be considered a special case of one-way-only, wherein both directions, but are not operative at the same time.) In a one-way-only processor bit slicing is not required (slice factor in (1) is one). For transmit, the timer is started when the start bit is output, and a bit is output at each interrupt. For receive, when a start bit is detected, the timer is set for one-half bit time. If the start bit is valid at that time, the timer is set for a full bit time, and one bit is shifted in at each interrupt. Since the amount of overhead processing is significantly reduced, higher baud rates may be realized with one-way-only processing.

Code Conversions

It may often be convenient or necessary to include code conversions in microprocessor firmware. For instance, an ASCII device might be required to communicate with an EBCDIC system. Or the designer may wish his device to be able to work with any of several codes. In the software design, a decision must be made as to where to do code conversion - in MAIN, or in the UART. In general, if the conversion process is lengthy, it should probably be in MAIN. However, if the code used is dependent upon data link parameters (such as baud rate), the UART might be a more logical place for the conversion. In figures 10 and 13 the code conversions appear as optional processes in the UART. In the example mentioned earlier, data for the two slower baud rates was converted to the 5 bit Baudot code by the UART.

The Real Thing

Figure 15 is a photo of the Mostek VAB-2 (Video Adaptor Board) CRT terminal electronics, built around an MK 3870 single-chip microcomputer. The VAB-2 utilizes the software UART described above.

ROM bytes required in the system:

Initialization	32	
UART	315	
MAIN	357	
Unused	1344	(65.6%)
	2048	

CPU registers required:

UART	11	
MAIN	8	
Unused	45	(70.3%)
	64	

Port bits used:

UART	4
MAIN	26
Unused	3
	33

CONCLUSION

A single-chip microprocessor such as the MK 3870 makes an ideal circuit element, eliminating much hardware cost while increasing performance, versatility, and ease of upgrade.

REFERENCES

1. Michael S. Miller, "Design a low-cost CRT terminal around a single-chip μP", EDN, 5 May 1977
2. John B. Peatman, Microcomputer Based Design, McGraw Hill 1977, pp 288-291
3. EIA Standard RS-404, March 1973
4. EIA Standard RS-232-C, August 1969
5. MK 3870 Data Sheet, Mostek Corporation, August 1977
6. VAB-2 Operations Manual, Mostek Corporation, August 1977
7. VAB-2 Program Source Listing, Mostek Corporation, April 1977

MOSTEK®
Z80·F8 Covering the full spectrum of
3870 microcomputer applications.

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/242-0444
In Europe, contact: MOSTEK GmbH, Talstrasse 172
D 7024 Filderstadt-1, W. Germany • Tele: (0711) 701096

PRINTED IN USA December 1977
Publication No. MK 79560